

# CS 466/666 Algorithm Design and Analysis

Keven Qiu

# Contents

<b>I</b>	<b>Amortized Analysis</b>	<b>5</b>
<b>1</b>	<b>Amortized Analysis</b>	<b>6</b>
1.1	Introduction . . . . .	6
1.2	Splay Trees . . . . .	8
1.2.1	Splay Operations . . . . .	8
1.2.2	Splay Tree Algorithm . . . . .	9
1.2.3	Analysis . . . . .	9
<b>II</b>	<b>Randomized Algorithms</b>	<b>13</b>
<b>2</b>	<b>Concentration Inequalities</b>	<b>14</b>
2.1	Markov's Inequality . . . . .	14
2.2	Chebyshev's Inequality . . . . .	15
2.3	Chernoff Bounds . . . . .	17
2.4	Hoeffding's Inequality . . . . .	19
<b>3</b>	<b>Balls and Bins</b>	<b>20</b>
3.1	Introduction . . . . .	20
3.1.1	Expected Number of Balls in a Bin . . . . .	21
3.1.2	Expected Number of Empty Bins . . . . .	22
3.1.3	Maximum Load in a Bin . . . . .	22
3.1.4	Maximum Load in a Bin when $m = n$ . . . . .	23
3.2	Coupon Collector and Power of Two Choices . . . . .	23

3.2.1	Coupon Collector . . . . .	23
3.2.2	Power of Two Choices . . . . .	24
<b>4</b>	<b>Hashing</b>	<b>25</b>
4.1	Hash Functions . . . . .	25
4.1.1	Computational Model . . . . .	25
4.1.2	Hash Functions . . . . .	25
4.1.3	Random Hash Functions . . . . .	26
4.2	$k$ -Wise Independence . . . . .	27
4.3	Universal Hash Functions . . . . .	28
4.4	Perfect Hashing . . . . .	30
<b>5</b>	<b>Graph Sparsification</b>	<b>31</b>
5.1	Minimum Cut . . . . .	31
5.2	Graph Sparsification Algorithm . . . . .	33
<b>6</b>	<b>Algebraic Techniques: Fingerprinting, Polynomial Identity Testing, and Parallel Matching Algorithms</b>	<b>36</b>
6.1	Verifying String Equality . . . . .	36
6.2	Polynomial Identity Testing . . . . .	37
6.3	Bipartite Matching . . . . .	38
6.4	General Matching . . . . .	38
6.5	Parallel Algorithms and Isolation Lemma . . . . .	39
<b>7</b>	<b>Sublinear Time Algorithms</b>	<b>41</b>
7.1	Approximate Diameter . . . . .	41
7.1.1	Lower Bound . . . . .	42
7.2	Connected Components . . . . .	42
<b>8</b>	<b>Random Walks and Markov Chains</b>	<b>44</b>
8.1	Random Walks . . . . .	44
8.2	Markov Chains . . . . .	45

8.3	Stationary Distributions . . . . .	46
8.3.1	Mixing Time . . . . .	47
8.4	Hitting Time . . . . .	48
8.5	Linear Algebra Background . . . . .	48
8.5.1	Perron-Frobenius . . . . .	49
8.6	Fundamental Theorem of Markov Chains . . . . .	51
8.7	Page Rank . . . . .	52
<b>III</b>	<b>Mathematical Programming</b>	<b>54</b>
<b>9</b>	<b>Linear Programming</b>	<b>55</b>
9.1	Introduction . . . . .	55
9.2	Fundamental Theorem of Linear Inequalities . . . . .	56
9.3	Duality Theory . . . . .	57
9.3.1	Complementary Slackness . . . . .	59
9.4	Applications of LP Duality . . . . .	59
9.4.1	Game Theory - Minimax Theorems . . . . .	59
9.4.2	Learning Theory - Boosting . . . . .	62
9.4.3	Combinatorics - Bipartite Matching . . . . .	63
<b>10</b>	<b>Semidefinite Programming</b>	<b>64</b>
10.1	Positive Semidefinite Matrices . . . . .	64
10.2	Semidefinite Programming Formulation . . . . .	65
10.2.1	LP vs. SDP . . . . .	65
10.3	Convex Algebraic Geometry . . . . .	66
10.3.1	Testing Points in Spectrahedron . . . . .	67
10.4	Control Theory . . . . .	68
10.5	Duality Theory . . . . .	69

<b>IV</b>	<b>Approximation Algorithms</b>	<b>72</b>
<b>11</b>	<b>Linear Programming Relaxations and Rounding</b>	<b>73</b>
11.1	Independent Set . . . . .	73
11.2	Vertex Cover . . . . .	75
11.2.1	Unweighted Simple 2-Approximation . . . . .	75
11.2.2	LP Relaxation . . . . .	75
11.3	Set Cover . . . . .	76
11.3.1	LP Relaxation . . . . .	76
11.3.2	Random Pick . . . . .	76
11.4	Randomized Rounding . . . . .	77
11.4.1	Cost of Rounding . . . . .	78
<b>12</b>	<b>Semidefinite Programming Relaxations and Rounding</b>	<b>79</b>
12.1	Maximum Cut . . . . .	79
12.1.1	SDP Explanation . . . . .	81
12.1.2	Rounding . . . . .	82
<b>V</b>	<b>Online Algorithms</b>	<b>84</b>
<b>13</b>	<b>Online Algorithms and Paging</b>	<b>85</b>
13.1	Ski Rental Problem . . . . .	85
13.2	Dating Problem . . . . .	86
13.3	Online Paging Problem . . . . .	87
13.3.1	Heuristics . . . . .	88
<b>14</b>	<b>Multiplicative Weights Update Method</b>	<b>90</b>

# Part I

## Amortized Analysis

# Chapter 1

## Amortized Analysis

### 1.1 Introduction

#### Definition: Amortized Analysis

Average of the total time required to perform a sequence of data-structure operations over all operations performed.

Amortized analysis is a **worst-case** analysis.

#### Types of Amortized Analysis

1. **Aggregate Analysis:** Upper bound  $T(n)$  on total cost of sequence of  $n$  operations. Amortized complexity is  $T(n)/n$ .
2. **Accounting Method:** Assign certain charges to each operation. If the operation is cheaper than the charge, then build up credit to use later.
3. **Potential Method:** A potential energy of a data structure, which maps each state of the entire data structure to  $\mathbb{R}$  (potential). Assign credit to the whole data structure instead of each operation in accounting method.

#### Problem

**Input:** A binary counter  $C$  initially set to 0.

**Output:** Increment this counter up to  $n$ .

**Question:** How many bit operations will it take to increment  $C$  from 0 to  $n$ ?

**Aggregate Analysis:** The worst-case time per operation is  $\log n$ . So an upper bound is  $O(n \log n)$ .

The most significant bits get updated very infrequently. Flip the  $k$ th bit after  $2^{k-1}$  opera-

tions/increments.

$$\# \text{ bit flips} = \sum_{k=0}^{\lceil \log n \rceil} \left\lfloor \frac{n}{2^k} \right\rfloor < \sum_{k=0}^{\infty} \frac{n}{2^k} = 2n$$

The total time is  $\Theta(n)$ . This implies that the amortized cost per operation is  $\Theta(1)$ .

**Accounting Method:** Suppose the actual cost of each operation of an algorithm is  $c_i$ . In each step of the algorithm, we assign charges  $\gamma_i$  to each operation such that

$$\sum_{i=1}^l \gamma_i \geq \sum_{i=1}^l c_i$$

for any  $l \geq 1$ . That is, the total charged up to step  $l$  is greater than or equal to the actual cost of all operations up to that point.

Suppose we charge the cost of clearing a bit ( $1 \rightarrow 0$ ) to the operation that sets the bit to 1 in the first place. If we flip  $k$  bits during an increment, we have already charged  $k - 1$  of those bit flips to earlier bit flips. Note that if we flip  $k$  bits, we must set  $k - 1$  of these bits to 0, so that it carries over.

So instead of paying for  $k$  bit flips, we charge at most 2: one for setting a bit to 1 and the other is to clear this bit.

So then total cost  $\leq$  total charged  $= 2n$ .

**Potential Method:** Suppose the actual cost of each operation of our algorithm is  $c_i$ . Assign potential  $\Phi_i$  to data structure at time  $i$ . Amortized cost of  $i$ th operation is

$$\gamma_i = c_i + \Phi_i - \Phi_{i-1}$$

That is, total amortized cost is the actual cost of the operation plus the change in potential. We have

$$\sum_{i=1}^n \gamma_i = \sum_{i=1}^n (c_i + \Phi_i - \Phi_{i-1}) = \Phi_n - \Phi_0 + \sum_{i=1}^n c_i$$

So if  $\Phi_k - \Phi_0 \geq 0$  for all  $k \geq 0$ , the total amortized cost is an upper bound on total cost.

Potential:  $\Phi_i$  = number of bits with value 1 at step  $i$  for  $i \geq 0$ . Now for every  $k \geq 0$ ,  $\Phi_k \geq \Phi_0 = 0$ .

$c_i = (\# \text{ bits } 0 \rightarrow 1) + (\# \text{ bits } 1 \rightarrow 0)$ .

$\Phi_i - \Phi_{i-1} = (\# \text{ bits } 0 \rightarrow 1) - (\# \text{ bits } 1 \rightarrow 0)$ .

Amortized cost:

$$\gamma_i = c_i + \Phi_i - \Phi_{i-1} = 2 \times (\# \text{ bits } 0 \rightarrow 1) = 2$$

since each increment only changes 1 bit from 0 to 1, each amortized cost is 2. So,

$$\sum_{i=1}^n c_i + \Phi_n - \Phi_0 = \sum_{i=1}^n \gamma_i = \sum_{i=1}^n 2 = 2n$$



## 1.2 Splay Trees

Data structures with great amortized running time are great for internal processes, such as internal graph algorithms (e.g. MST). It is bad when you have client-server model, as in this setting, one wants to minimize worst-case per query.

### Definition: Splay Trees

Self-adjusting binary search trees.

### Theorem (Sleator & Tarjan 1985)

Splay trees have  $\Theta(\log n)$  amortized cost per operation.,  $\Theta(n)$  worst-case time.

Does not keep any balancing information. Adjust the tree whenever a node is accessed.

### Definition: Splaying

Move the node that was searched to the root.

### Notation

$n \leftarrow$  number of elements,  $m \leftarrow$  number of operations = searches + insertions + deletions.

Operations:  $SEARCH(k)$ ,  $INSERT(k)$ ,  $DELETE(k)$

### 1.2.1 Splay Operations

#### Definition: Splay Operation

$SPLAY(k)$

**Input:** element  $k$

**Output:** rebalancing of binary search tree

#### Definition: Zig-Zag Condition

$parent(k)$  has  $k$  as left/right-child and  $parent(parent(k))$  has  $parent(k)$  as right/left-child.

#### Definition: Zig-Zig Condition

$parent(k)$  has  $k$  as left/right-child and  $parent(parent(k))$  has  $parent(k)$  as left/right-child.

Only apply zig rotation when there is no grandparent of the node we are rotating.

---

**Algorithm 1** *SPLAY*( $k$ )

---

```
1: while  $k$  is not root do
2:   if  $k$  satisfies zig-zag condition then
3:     zig-zag rotation
4:   if  $k$  satisfies zig-zig condition then
5:     zig-zig rotation
6:   if  $k$  is a child of root then
7:     zig rotation (normal)
```

---

### 1.2.2 Splay Tree Algorithm

*SEARCH*( $k$ ): after searching for  $k$ , if  $k$  in the tree, do *SPLAY*( $k$ ). If  $k$  is not in the tree, then perform *SPLAY*( $k'$ ) where  $k'$  is the last key that was compared to  $k$  when searching.

*INSERT*( $k$ ): standard insert operation, then do *SPLAY*( $k$ )

*DELETE*( $k$ ): standard delete operation, then *SPLAY*(*parent*( $k$ ))

- delete first (moves  $k$  to the bottom of tree) by finding successor
- then delete  $k$  as in the cases where  $k$  has at most one child
- then we splay the parent of  $k$  (after placing  $k$  at the bottom)

Intuition: zig-zag and zig-zig make a lot of progress in unbalanced trees. If the tree is balanced, then splaying is quite fast.

### 1.2.3 Analysis

**Potential Method:** The charge  $\hat{c}_i$  of the  $i$ th operation with respect to the potential function  $\Phi$  is:

$$\hat{c}_i := c_i + \Phi(D_i) - \Phi(D_{i-1})$$

The amortized cost of all operations is

$$\begin{aligned} \sum_{i=1}^m \hat{c}_i &= \sum_{i=1}^m c_i + \Phi(D_i) - \Phi(D_{i-1}) \\ &= \Phi(D_m) - \Phi(D_0) + \sum_{i=1}^m c_i \geq \sum_{i=1}^m c_i \end{aligned}$$

So long as  $\Phi(D_m) \geq \Phi(D_0)$ , then amortized charge is an upper bound on amortized cost.

**Definition: Potential Function of Splay Tree**

- $\delta(k) :=$  number of descendants of  $k$  (including  $k$ )
- $\text{rank}(k) := \log(\delta(k))$
- 

$$\Phi(T) = \sum_{k \in T} \text{rank}(k)$$

The minimum potential is with a perfectly balanced tree. The root has rank  $\log n$ , any node in second level has rank  $\log \frac{n}{2}$ , and so forth. So

$$\Phi(T) = \sum_{h=1}^{\log n} h \cdot \frac{n}{2^h} = \Theta(n)$$

**Analysis - Splay Operation:** Let  $\text{rank}(k)$  be the current rank of  $k$  and  $\text{rank}'(k)$  be the new rank of  $k$  after we perform a rotation on  $k$ .

**Lemma (Amortized Cost from *SPLAY* Subroutines)**

The charge  $\gamma$  of an operation (zig, zig-zig, zig-zag) is bounded by:

$$\gamma \leq \begin{cases} 3(\text{rank}'(k) - \text{rank}(k)) & \text{for zig-zig, zig-zag} \\ 3(\text{rank}'(k) - \text{rank}(k)) + 1 & \text{for zig} \end{cases}$$

**Proof.** Let  $T'$  be the tree after rotation.

We begin by analyzing the zig rotation. Let  $k$  be the node we rotating,  $b = \text{parent}(k)$ . Then,  $\text{rank}'(k) = \text{rank}(b)$ . The charge in this case is given by

$$\begin{aligned} \gamma &= \text{cost} + \Phi(T') - \Phi(T) \\ &= 1 + \text{rank}'(k) + \text{rank}(b) - \text{rank}(k) - \text{rank}(b) \\ &= 1 + \text{rank}'(b) - \text{rank}(k) \\ &\leq 1 + \text{rank}'(k) - \text{rank}(k) \\ &\leq 1 + 3(\text{rank}'(k) - \text{rank}(k)) \end{aligned}$$

where  $\text{rank}'(b) \leq \text{rank}'(k)$  since  $b$  is a child of  $k$  in  $T'$ .

Analyzing the zig-zag rotation. Let  $k$  be the node we are rotating,  $b = \text{parent}(k)$ , and  $a = \text{parent}(b)$ . Then,  $\text{rank}'(k) = \text{rank}(a)$ ,  $\text{rank}'(b) \leq \text{rank}'(k)$ , and  $\text{rank}(k) \leq \text{rank}(b) \leq \text{rank}(a)$ . Moreover,  $\delta'(k) \geq \delta'(a) + \delta(k)$ . The charge in this case is given by

$$\begin{aligned} \gamma &= \text{cost} + \Phi(T') - \Phi(T) \\ &= 2 + \text{rank}'(a) + \text{rank}'(b) + \text{rank}'(k) - \text{rank}(a) - \text{rank}(b) - \text{rank}(k) \\ &= 2 + \text{rank}'(a) + \text{rank}'(b) - \text{rank}(b) - \text{rank}(k) \\ &\leq 2 + \text{rank}'(a) + \text{rank}'(k) - 2 \cdot \text{rank}(k) \end{aligned}$$

Now since  $\delta'(k) \geq \delta'(a) + \delta(k)$ , by concavity of  $\log$ , we have

$$\log \left( \frac{\delta'(a)}{\delta'(k)} \right) + \log \left( \frac{\delta(k)}{\delta'(k)} \right) \leq -2 \implies \log \delta'(a) + \log \delta(k) \leq 2 \log \delta'(k) - 2$$

Equivalently, we have  $\text{rank}'(a) \leq 2 \cdot \text{rank}'(k) - \text{rank}(k) - 2$ . Thus, we have

$$\gamma \leq 2 + \text{rank}'(a) + \text{rank}'(k) - 2 \cdot \text{rank}(k) \leq 3(\text{rank}'(k) - \text{rank}(k))$$

The proof for zig-zig is similar to zig-zag.

**Lemma (Total Amortized Cost of  $SPLAY(k)$ )**

Let  $T$  be our current tree, with root  $t$  and  $k$  be a node in this tree. The charge  $\Gamma$  of the  $SPLAY(k)$  operation is bounded by:

$$\Gamma \leq 3(\text{rank}(t) - \text{rank}(k)) + 1 \leq 3\text{rank}(t) + 1 = O(\log n)$$

**Proof.** Note that  $\Gamma$  is the sum of the charges of the basic rotations performed during the  $SPLAY(k)$  operation. Let  $\gamma_i$  be the charge of the  $i$ th rotation, and  $\text{rank}_i(k)$  be the rank of  $k$  after the  $i$ th rotation. Then, we have  $\text{rank}_0(k) = \text{rank}(k)$ ,  $\text{rank}_l(k) = \text{rank}(t)$ , where  $l$  is the number of basic rotations performed during the  $SPLAY(k)$  operation.

There is only one zig operation so we add 1. Thus,

$$\Gamma = \sum_{i=1}^l \gamma_i \leq 1 + \sum_{i=1}^l 3(\text{rank}_i(k) - \text{rank}_{i-1}(k)) = 1 + 3(\text{rank}(t) - \text{rank}(k))$$

where the inequality comes from the previous lemma.

**Analysis - Amortized Cost:** For each of the 3 operations we have:

$$\text{charge per operation} = (\text{charge of } SPLAY) + (\text{potential change not from } SPLAY)$$

The charge of  $SPLAY$  is  $O(\log n)$  from second lemma. Charge of  $SPLAY$  already includes the cost of the operation.

Tracking potential outside splay:

- *SEARCH*: only splay changes the potential
- *DELETE*: removing a node decreases potential
- *INSERT*: adding new element  $k$  increases ranks of all ancestors of  $k$  post insertion (might be  $O(n)$  of them)

So we need to handle case 3: Let  $k := k_0 \mapsto k_1 \mapsto \dots \mapsto k_l$  where  $k - I$  is the  $i$ th ancestor of  $k$  and  $k_l$  is the root of the tree. If we denote  $\delta'(a)$  as the number of descendants of  $a$

post insertion, then we have  $\delta'(k) = 1$  (since before splaying,  $k$  is a leaf of the tree) and  $\delta'(k_i) = \delta(k_i) + 1$  for  $1 \leq i \leq l$ . Hence, the change in potential is:

$$\sum_{i=1}^l \log \left( \frac{\delta'(k_i)}{\delta(k_i)} \right) = \sum_{i=1}^l \log \left( \frac{\delta(k_i) + 1}{\delta(k_i)} \right) \leq \sum_{i=1}^n \log \left( \frac{i+1}{i} \right) = \log(n+1) = O(\log n)$$

Thus, the amortized cost of each operation is  $O(\log n)$ , since we upper bounded each of the 3 quantities (charge of splay, cost of operation, potential change not from splay) by  $O(\log n)$ .

To show our potential function is valid, the initial potential is 0 for the empty tree and the potential is always nonnegative (sum of logarithms).

### Dynamic Optimality Conjecture (Sleator & Tarjan 1985)

Splay trees are optimal within a constant in a very strong sense:

Given a sequence of items to search for  $a_1, \dots, a_m$ , let  $OPT$  be the minimum cost of doing these searches + any rotations you like on the binary search tree.

You can charge 1 for following tree pointer (parent  $\rightarrow$  child or child  $\rightarrow$  parent), charge 1 per rotation.

Conjecture: Cost of splay tree is  $O(OPT)$ .

## Part II

# Randomized Algorithms

# Chapter 2

## Concentration Inequalities

When evaluating the performance of randomized algorithms, we want to not only analyze the expected runtimes, but also if the algorithm runs in time close to its expected runtime *most of the time*.

A small runtime with high probability is *better than* small expected runtime.

### 2.1 Markov's Inequality

#### Theorem (Markov's Inequality)

Let  $X$  be a nonnegative discrete random variable. Then

$$P[X \geq t] \leq \frac{\mathbb{E}[X]}{t}, \forall t > 0$$

**Proof.**

$$\begin{aligned} \mathbb{E}[X] &= \sum_{n \geq 0} P[X = n] \cdot n \\ &= \sum_{n=0}^{t-1} P[X = n] \cdot n + \sum_{n \geq t} P[X = n] \cdot n \\ &\geq 0 + t \cdot \sum_{n \geq t} P[X = n] \\ &= t \cdot P[X \geq t] \end{aligned}$$

**Quicksort:** The expected runtime of quicksort is  $2n \log n$ . Markov's inequality tells us that the runtime is at least  $2cn \log n$  with probability  $\leq 1/c$ , for any  $c \geq 1$ .

**Coin Flipping:** If we flip  $n$  fair coins, the expected number of heads is  $n/2$ . Markov's inequality tells that  $P[\# \text{ heads} \geq 3n/4] \leq 2/3$ .

Remark: Useful when there is no information beyond expected value.

## 2.2 Chebyshev's Inequality

### Definition: Variance

$$\text{Var}[X] := \mathbb{E}[(X - \mathbb{E}[X])^2] = \mathbb{E}[X^2] - \mathbb{E}[X]^2$$

### Definition: Standard Deviation

$$\sigma(X) := \sqrt{\text{Var}[X]}$$

### Theorem (Chebyshev's Inequality)

Let  $X$  be a discrete random variable. Then

$$P[|X - \mathbb{E}[X]| \geq t] \leq \frac{\text{Var}[X]}{t^2}, \quad \forall t > 0$$

**Proof.** We only know Markov's inequality. Let  $Y = (X - \mathbb{E}[X])^2$ . Then,

$$Y \begin{cases} \text{discrete} & \text{if } X \text{ discrete} \\ \geq 0 \end{cases}$$

so we can use Markov's inequality.

$$P[Y \geq t^2] = P[|X - \mathbb{E}[X]| \geq t] \leq \frac{\mathbb{E}[Y]}{t^2} = \frac{\text{Var}[X]}{t^2}$$

### Definition: Covariance

The covariance of two random variables  $X, Y$  is defined as

$$\text{Cov}[X, Y] := \mathbb{E}[(X - \mathbb{E}[X])(Y - \mathbb{E}[Y])] = \mathbb{E}[XY] - \mathbb{E}[X]\mathbb{E}[Y]$$

Note that  $\text{Cov}[X, X] = \text{Var}[X]$ .

We say that  $X, Y$  are positively correlated if  $\text{Cov}[X, Y] > 0$  and negatively correlated if  $\text{Cov}[X, Y] < 0$ .

Independent random variables are uncorrelated, but uncorrelated random variables are not necessarily independent.



**Proposition**

- $Var[X + Y] = Var[X] + Var[Y] + 2Cov[X, Y]$
- If  $X, Y$  are independent, then  $Var[X + Y] = Var[X] + Var[Y]$

**Coin Flipping:** If  $X$  be # heads in  $n$  independent unbiased flips, let us bound  $P[X \geq 3n/4]$ .

$$X_i = \begin{cases} 1 & \text{if heads} \\ 0 & \text{otherwise} \end{cases}$$

Then,  $X = \sum_{i=1}^n X_i$  and each  $X_i, X_j$  are independent.

$$\mathbb{E}[X_i] = 1 \cdot \frac{1}{2} + 0 \cdot \frac{1}{2} = \frac{1}{2}$$

and

$$Var[X_i] = \mathbb{E}[(X_i - \mathbb{E}[X_i])^2] = \frac{1}{2} \left(1 - \frac{1}{2}\right)^2 + \frac{1}{2} \left(0 - \frac{1}{2}\right)^2 = \frac{1}{4}$$

So by proposition,

$$Var[X] = \sum_{i=1}^n Var[X_i] = \frac{n}{4}$$

By Chebyshev's inequality,

$$P\left[X \geq \frac{3n}{4}\right] \leq P\left[\left|X - \frac{n}{2}\right| \geq \frac{n}{4}\right] \leq \frac{n/4}{(n/4)^2} = \frac{4}{n}$$

Chebyshev's inequality is most useful when we only have information about the second moment of  $X$ .

**Definition:  $k$ th Moment**

The  $k$ th moment of a random variable  $X$  is  $\mathbb{E}[X^k]$ .

**Definition:  $k$  Central Moment**

The  $k$  central moment of a random variable  $X$  is

$$\mu_X^{(k)} := \mathbb{E}[(X - \mathbb{E}[X])^k]$$

if it exists.

**Definition: i.i.d.**

Independent and identically distributed.

**Definition: Law of Large Numbers**

Average of i.i.d. variables is approximately the expectation of the random variables.

$$\frac{1}{n} \cdot \sum_{i=1}^n X_i \approx \mathbb{E}[X]$$

## 2.3 Chernoff Bounds

**Definition: Chernoff Bounds**

Give quantitative estimates of the probability that  $X$  is far from  $\mathbb{E}[X]$  for any value of  $n$ , when  $X = \sum_{i=1}^n X_i$ .

**Definition: Moment Generating Function**

$$M_X(t) := \mathbb{E}[e^{tX}] = \mathbb{E} \left[ \sum_{k \geq 0} \frac{t^k}{k!} \cdot X^k \right] = \sum_{k \geq 0} \frac{t^k \mathbb{E}[X^k]}{k!}$$

**Theorem (Chernoff Inequality)**

Let  $X_1, \dots, X_n$  be independent variables such that  $X_i \in 0, 1$  for all  $i \in [n]$ . Let  $X = \sum_{i=1}^n X_i$  and  $\mu = \mathbb{E}[X]$ . Then, for  $\delta > 0$ ,

$$P[X \geq (1 + \delta)\mu] \leq \left[ \frac{e^\delta}{(1 + \delta)^{1+\delta}} \right]^\mu$$

and

$$P[X \leq (1 - \delta)\mu] \leq \left[ \frac{e^{-\delta}}{(1 - \delta)^{1-\delta}} \right]^\mu$$

**Theorem (Chernoff Inequality -  $0 < \delta < 1$ )**

Let  $X_1, \dots, X_n$  be independent variables such that  $X_i \in 0, 1$  for all  $i \in [n]$ . Let  $X = \sum_{i=1}^n X_i$  and  $\mu = \mathbb{E}[X]$ . Then, for  $0 < \delta < 1$ ,

$$P[X \geq (1 + \delta)\mu] \leq e^{-\mu\delta^2/3}$$

and

$$P[X \leq (1 - \delta)\mu] \leq e^{-\mu\delta^2/2}$$

**Proof.** We will prove the first inequality. Let  $p_i := P[X_i = 1]$  and thus,  $P[X_i = 0] = 1 - p_i$  and  $\mu = \sum_{i=1}^n p_i$ .

Idea is to use Markov's inequality to the random variable  $e^{tX}$ . Since the exponential function is increasing, we have

$$P[X \geq a] = P[e^{tX} \geq e^{ta}] \leq \frac{\mathbb{E}[e^{tX}]}{e^{ta}}, \quad \forall t > 0$$

When we use the exponential function, we are using information about all moments of  $X$ . The Taylor series of  $e^{tX}$  is

$$e^{tX} = \sum_{k \geq 0} \frac{(tX)^k}{k!} = 1 + tX + \frac{t^2 X^2}{2!} + \frac{t^3 X^3}{3!} + \dots$$

If  $X = X_1 + X_2$  where  $X_1, X_2$  are independent, then  $M_X(t) = M_{X_1}(t)M_{X_2}(t)$ .

Apply Markov's inequality to  $e^{tX}$ .

$$P[X \geq (1 + \delta)\mu] = P[e^{tX} \geq e^{t(1+\delta)\mu}] \leq \frac{\mathbb{E}[e^{tX}]}{e^{t(1+\delta)\mu}}$$

By the above and independence of  $X_i$ 's, we have

$$\mathbb{E}[e^{tX}] = \prod_{i=1}^n \mathbb{E}[e^{tX_i}] = \prod_{i=1}^n (p_i \cdot e^t + (1 - p_i) \cdot 1)$$

Since  $p_i \cdot e^t + (1 - p_i) \cdot 1 = 1 + p_i \cdot (e^t - 1) \leq e^{p_i(e^t - 1)}$  as  $e^x \geq 1 + x$  for all  $x \geq 0$ , we have

$$\frac{\mathbb{E}[e^{tX}]}{e^{t(1+\delta)\mu}} \leq \frac{\prod_{i=1}^n e^{p_i(e^t - 1)}}{e^{t(1+\delta)\mu}} = \left( \frac{e^{e^t - 1}}{e^{t(1+\delta)}} \right)^\mu \leq \left( \frac{e^\delta}{(1 + \delta)^{1+\delta}} \right)^\mu$$

where  $t = \ln(1 + \delta)$ . The main inequality follows and the fact that  $\frac{e^\delta}{(1 + \delta)^{1+\delta}} \leq e^{-\delta^2/3}$  for all  $0 < \delta < 1$ .

## 2.4 Hoeffding's Inequality

### Lemma (Hoeffding)

If  $Z$  is a random variable such that  $Z \in [a, b]$ , then

$$\mathbb{E}[e^{t(Z - \mathbb{E}[Z])}] \leq e^{t^2(b-a)^2/8}$$

### Theorem (Hoeffding's Inequality)

Let  $X_i$  be independent random variables, taking values in  $[a_i, b_i]$ . Let  $X = \sum_{i=1}^n X_i$  and  $\mu = \mathbb{E}[X]$ . Then, for any  $l > 0$

$$P[|X - \mu| \geq l] \leq 2 \cdot \exp\left(-\frac{2l^2}{\sum_{i=1}^n (b_i - a_i)^2}\right)$$

**Proof.** Similar to Chernoff's inequality, but use Hoeffding's lemma.

# Chapter 3

## Balls and Bins

### 3.1 Introduction

#### Theorem (Union Bound)

Can be generalized to any level of intersections.

$$P[A \cup B \cup C] \leq P[A] + P[B] + P[C]$$

$$P[A \cup B \cup C] \geq P[A] + P[B] + P[C] - P[A \cap B] - P[A \cap C] - P[B \cap C]$$

#### Definition: Conditional Probability

The conditional probability of  $E_1$  given  $E_2$  is

$$P[E_1|E_2] := \frac{P[E_1 \cap E_2]}{P[E_2]}$$

#### Proposition

If  $E_1, \dots, E_k$  partition out sample space, then for any event  $E$

$$P[E] = \sum_{i=1}^k \underbrace{P[E|E_i] \cdot P[E_i]}_{P[E \cap E_i]}$$

#### Theorem (Simple Bayes' Rule)

$$P[E_1|E_2] = \frac{P[E_2|E_1] \cdot P[E_1]}{P[E_2]}$$

### Theorem (Bayes' Rule)

If  $E_1, \dots, E_k$  partition our sample space, then for event  $E$

$$P[E_i|E] = \frac{P[E \cap E_i]}{P[E]} = \frac{P[E|E_i] \cdot P[E_i]}{\sum_{j=1}^k P[E|E_j] \cdot P[E_j]}$$

### Balls and Bins

Given  $m$  balls and  $n$  bins to throw each ball into a uniformly random bin independently.

- What is the expected number of balls in a bin?
- What is the expected number of empty bins?
- What is typically the maximum number of balls in any bin (maximum load)?
- What is the expected number of bins with  $k$  balls in them?
- For what values of  $m$  do we expect to have no empty bins? (coupon collector)

**Strategy of Randomized Algorithms:** Devise the randomized algorithm with good expected runtime and prove concentration of measure around expectation.

#### 3.1.1 Expected Number of Balls in a Bin

Label the  $m$  balls  $1, \dots, m$  and the  $n$  bins  $1, \dots, n$ . Let  $B_{ij}$  be the indicator variables that ball  $i$  was thrown into bin  $j$ .

$$\begin{aligned}\mathbb{E}[\# \text{ balls in bin } j] &= \mathbb{E}\left[\sum_{i=1}^m B_{ij}\right] \\ &= \sum_{i=1}^m \mathbb{E}[B_{ij}] \\ &= \sum_{i=1}^m 1 \cdot P[\text{ball } i \text{ in bin } j] + 0 \cdot (1 - P[\text{ball } i \text{ in bin } j]) \\ &= \sum_{i=1}^m P[\text{ball } i \text{ in bin } j] \\ &= \sum_{i=1}^m \frac{1}{n} = \frac{m}{n}\end{aligned}$$

When  $m = n$ , expect one ball per bin.

### 3.1.2 Expected Number of Empty Bins

Let  $N_i$  be the indicator variable that bin  $i$  is empty after  $m$  throws.

$$\begin{aligned}
\mathbb{E}[\# \text{ empty bins}] &= \mathbb{E} \left[ \sum_{i=1}^n N_i \right] \\
&= \sum_{i=1}^n \mathbb{E}[N_i] \\
&= \sum_{i=1}^n P[\text{bin } i \text{ is empty}] \\
&= \sum_{i=1}^n \left(1 - \frac{1}{n}\right)^m \\
&= n \left(1 - \frac{1}{n}\right)^m \\
&\approx ne^{-m/n} \qquad \left(1 - \frac{1}{n}\right)^n \approx e^{-1}
\end{aligned}$$

When  $m = n$ , expected fraction of empty bins is  $\frac{1}{e}$ .

When  $m = n$ , the first expectation was one ball per bin while the second expectation was  $\frac{1}{e}$  fraction of empty bins. This is where concentration of probability measure tries to address. The second expectation is actually concentrated around the mean so more typical to happen.

### 3.1.3 Maximum Load in a Bin

<b>Birthday Paradox</b>
For what value of $m$ do we expect to see two balls in one bin?

The probability that there are no collisions after we have thrown  $m$  balls is:

$$1 \left(1 - \frac{1}{n}\right) \left(1 - \frac{2}{n}\right) \cdots \left(1 - \frac{m-1}{n}\right) \leq e^{-\frac{1}{n}} \cdots e^{-\frac{m-1}{n}} \approx e^{-m^2/2n}$$

This is  $\leq 1/2$  when  $m = \sqrt{2n \ln 2}$ . For  $n = 365$ , this is  $m \approx 22.4$  for the the probability that two people (balls) have birthday on the same date (bins) to become  $\geq 1/2$ .

Thus, we expect to see collision when  $m = \Theta(\sqrt{n})$ .

### 3.1.4 Maximum Load in a Bin when $m = n$

What is the probability that a particular bin has  $\geq k$  balls in it?

$$\begin{aligned}
 P[\text{bin } x \text{ has } \geq k \text{ balls}] &\leq \sum_{\substack{S \subseteq [n] \\ |S|=k}} \prod_{i \in S} P[\text{ball } i \text{ in bin } x] \\
 &= \sum_{\substack{S \subseteq [n] \\ |S|=k}} \prod_{i \in S} \frac{1}{n} \\
 &= \binom{n}{k} \cdot \frac{1}{n^k} \\
 &\leq \left(\frac{ne}{k}\right)^k \cdot \frac{1}{n^k} \\
 &= \frac{e^k}{k^k}
 \end{aligned}$$

By union bound,

$$P[\text{some bin has } \geq k \text{ balls}] \leq \sum_{i=1}^n P[\text{bin } i \text{ has } \geq k \text{ balls}] \leq \frac{ne^k}{k^k} = e^{\ln n + k - k \ln k}$$

Therefore, the probability of maximum load at most  $k$  is

$$P[\text{max load} \leq k] = 1 - P[\text{some bin has } > k \text{ balls}] \geq 1 - e^{\ln n + k - k \ln k}$$

The above probability will be large ( $\gg \frac{1}{2}$ ) when  $k \ln k > \ln n$  such as setting  $k = \frac{3 \ln n}{\ln \ln n}$ .

With high probability, the maximum load is  $O\left(\frac{\ln n}{\ln \ln n}\right)$ .

## 3.2 Coupon Collector and Power of Two Choices

### 3.2.1 Coupon Collector

**Coupon Collector:** For what value of  $m$  do we expect to have no empty bins?

Suppose each bin is a different coupon. We can buy one coupon at random. What is the number of coupons that we need to buy to collect all of them?

Let  $X_i$  be the number of balls thrown to get from  $i$  empty bins to  $i - 1$  empty bins. Let  $X$  be the number of balls thrown until we have no empty bins. Thus,  $X = \sum_{i=1}^n X_i$ .

$X_i$  is a random variable that follows a geometric distribution with parameter  $p = \frac{i}{n}$  (Geometric distribution is the number of trials until first success, where success probability is  $p$ ).

$$P[X_i = k] = p(1 - p)^{k-1}$$



So the expected value of  $X_i$  where  $X_i$  takes values over  $\mathbb{N}$  is

$$\mathbb{E}[X_i] = \sum_{k \geq 1} k \cdot P[X_i = k] = p_i \sum_{k \geq 1} (1-p)^{k-1} = p_i \cdot \frac{1}{(1-(1-p))^2} = \frac{1}{p} = \frac{n}{i}$$

by the derivative of the closed form of the series. Thus,

$$\mathbb{E}[X] = \sum_{i=1}^n \mathbb{E}[X_i] = \sum_{i=1}^n \frac{n}{i} = n \sum_{i=1}^n \frac{1}{i} \approx n \ln n$$

The  $n \ln n$  bound shows up in cover time of random walks in complete graphs and the number of edges needed in graph sparsification.

### 3.2.2 Power of Two Choices

We now know that when  $n$  balls are thrown into  $n$  bins, the maximum load is  $\Theta\left(\frac{\ln n}{\ln \ln n}\right)$  with constant probability.

Consider if when throwing a ball in a bin, before we throw the ball we choose two bins uniformly at random and put the ball in the bin with few balls.

This simplification reduces maximum load to  $O(\ln \ln n)$ .

*Idea:* Let the height of a bin be the number of balls in it. The process above tells us that to get one bin height  $h+1$ , we must have at least two bins with height  $h$ . We can bound the number of bins with height  $\geq h$ , as this will tell us how likely it is to get a bin with height  $h+1$ .

If  $N_h$  is the number of bins with height  $\geq h$ , then we have

$$P[\text{at least one bin of height } h+1] \leq \binom{N_h}{2} \leq \left(\frac{N_h}{n}\right)^2$$

To bound  $N_h$ , say we only have  $\frac{n}{4}$  bins with 4 items, i.e. height 4. Then, the probability of selecting 2 such bins is  $\leq \frac{1}{16}$ . So, we should expect only  $\frac{n}{16}$  bins with height 5. Analogously, we should expect only  $\frac{n}{16^2} = \frac{n}{256} = \frac{n}{2^{2^3}}$  bins with height 6.

Repeating this, we should expect only  $n/2^{2^{h-3}}$  bins with height  $h$ . So we expect  $\log \log n$  maximum height after throwing  $n$  balls.

# Chapter 4

## Hashing

### 4.1 Hash Functions

#### 4.1.1 Computational Model

**Definition: Word RAM Model**

In the word RAM model:

- All elements are integers that fit in a machine word of  $w$  bits.
- Basic operations (comparison, arithmetic, bitwise) on such words take  $\Theta(1)$  time.
- Access any position in the array in  $\Theta(1)$  time.

This model is relevant for problems of good enough size (so asymptotic analysis can work), but not super huge that words do not fit in a machine word.

#### 4.1.2 Hash Functions

**Problem**

Store  $n$  elements (keys) from the set  $U = \{0, 1, \dots, m - 1\}$  where  $2^w > m \gg n$ , in a data structure that supports insertion, deletions, search as efficiently in runtime and memory.

**Definition: Hash Function**

A function  $h : U \rightarrow [0, n - 1]$ , where  $|U| = m \gg n$ .

**Definition: Hash Table**

A data structure that consists of

- a table  $T$  with  $n$  cells  $[0, n - 1]$ , each cell storing a word
- a hash function  $h : U \rightarrow [0, n - 1]$

**Definition: Collision**

We say that a collision happens for hash function  $h$  with inputs  $x, y \in U$  if  $x \neq y$  and  $h(x) = h(y)$ .

By pigeonhole principle, it is impossible to achieve no collisions without knowing keys in advance. We want the number of collisions to be small *with high probability*.

**Solution:** Construct a family of hash function  $\mathcal{H}$  such that the number of collisions is small with high probability, when we pick hash function uniformly at random from  $\mathcal{H}$ .

$$P_{h \in \mathcal{H}}[h(x) = h(y)] \leq \frac{1}{\text{poly}(n)}, \quad \forall x \neq y \in U$$

Assumptions: keys are independent from hash function that are chosen and we do not know the keys in advance. This can still have collisions.

**4.1.3 Random Hash Functions**

From all functions  $h : U \rightarrow [0, n - 1]$ , pick one uniformly at random. This is the same as balls and bins.

If we have to store  $n$  keys:

- Expected number of keys in a location is 1.
- Maximum number of collisions (maximum load) in one location is  $O\left(\frac{\log n}{\log \log n}\right)$  keys.

To handle collisions, we can store all keys hashed into location  $i$  by a linked list, known as chain hashing.

We can also pick two random hash functions and use power of two choices. The collision bound becomes  $O(\log \log n)$ .

**Question:** How much time and space does it take to compute random hash functions?

Storing entire functions  $h : U \rightarrow [0, n - 1]$  requires  $O(m \log n)$  bits and if we only stored the elements we saw, we would require  $O(n)$  time to evaluate  $h(x)$ , since we need to decide if we had already computed it. Thus, for random function operations (search, insert, delete) take  $O(n)$  time at best.

## 4.2 $k$ -Wise Independence

We want something is random-like, but easy to compute/represent. Ideally,  $O(1)$  time to compute.  $h$  is described in  $c \log m$  bits. This is at most the number of bit strings of length  $c \log m$ :  $2^{c \log m} = m^c$  so  $\text{poly}(m)$  functions, as each function takes at most  $O(\log m)$  bits to describe. These are succinct functions which have random-like properties.

### Definition: Full Independence

A set of random variables  $X_1, \dots, X_n$  are said to be fully independent if for any subset  $J \subseteq [n]$ , they satisfy

$$P \left[ \bigcap_{i \in J} X_i = a_i \right] = \prod_{i \in J} P[X_i = a_i]$$

### Definition: $k$ -Wise Independence

A set of random variables  $X_1, \dots, X_n$  are said to be  $k$ -wise independent if for any set  $J \subseteq [n]$  such that  $|J| \leq k$ , they satisfy

$$P \left[ \bigcap_{i \in J} X_i = a_i \right] = \prod_{i \in J} P[X_i = a_i]$$

When  $k = 2$ ,  $k$ -wise independence is called pairwise independence.

### Example (XOR Pairwise Independence)

Given  $t$  uniformly random bits  $Y_1, \dots, Y_t$ , we can generate  $2^t - 1$  pairwise independent random variables as follows:

$$X_S := \bigoplus_{i \in S} Y_i, \quad S \subseteq [t] \setminus \emptyset$$

### Example (Pairwise Independence in $\mathbb{F}_p$ )

Let  $p$  be a prime number. Given two uniformly random variables  $Y_1, Y_2 \sim [0, \dots, p-1]$ , generate  $p$  pairwise independent random variables as follows:

$$X_i := Y_1 + i \cdot Y_2 \mod p, \quad i \in [0, p-1]$$

### 4.3 Universal Hash Functions

#### Definition: Universal Hash Functions

Let  $U$  be a universe with  $|U| \geq n$ . A family of hash functions  $\mathcal{H} = \{h : U \rightarrow [0, n-1]\}$  is  $k$ -universal if, for any distinct elements  $u_1, \dots, u_k \in U$ , we have

$$P_{h \in \mathcal{H}}[h(u_1) = h(u_2) = \dots = h(u_k)] \leq \frac{1}{n^{k-1}}$$

#### Definition: Strongly Universal Hash Functions

$\mathcal{H} = \{h : U \rightarrow [0, n-1]\}$  is strongly  $k$ -universal if, for any distinct elements  $u_1, \dots, u_k \in U$  and for any values  $y_1, \dots, y_k \in [0, n-1]$ , we have

$$P_{h \in \mathcal{H}}[h(u_1) = y_1, \dots, h(u_k) = y_k] \leq \frac{1}{n^k}$$

#### Theorem

Family  $\mathcal{H}$  is strongly  $k$ -universal if the random variables  $h(0), \dots, h(|U| - 1)$  are  $k$ -wise independent.

#### Proposition

Let  $p$  be a prime number and  $U = [0, p-1]$ .

$$\mathcal{H} = \{h_{a,b}(x) := ax + b \mod p \mid a, b \in [0, p-1]\}$$

is strongly 2-universal.

#### Proposition

Let  $U = [0, p^k - 1] \equiv [0, p-1]^k \setminus \{(0, \dots, 0)\}$  and  $a = (a_0, \dots, a_{k-1})$ , then

$$\mathcal{H} = \{h_{a,b}(x) := a \cdot x + b \mod p \mid a \in U, b \in [0, p-1]\}$$

is strongly 2-universal.

#### Proposition

If hash table size is not prime, then

$$\mathcal{H} = \{h_{a,b}(x) := (a \cdot x + b \mod p) \mod n \mid a, b \in [0, p-1]\}$$

is 2-universal (not strongly 2-universal).

We can construct  $k$ -universal families of hash functions by instead of constructing degree 1

polynomials, we can construct polynomials of degree  $k-1$ . Random  $k-1$  degree polynomial is  $k$ -wise independent.

**Lemma (Maximum Number of Collisions)**

The expected number of collisions when inserting  $k$  elements in a table of size  $n$  using a 2-universal hash family is

$$\leq \frac{k^2}{2n}$$

**Proof.** Let  $X_{ij} = 1$  if  $h(i) = h(j)$  and  $X_{ij} = 0$  otherwise. The number of collisions is  $X = \sum_{i < j} X_{ij}$ .

$$\begin{aligned} \mathbb{E}[X] &= \mathbb{E}\left[\sum_{i < j} X_{ij}\right] \\ &= \sum_{i < j} \mathbb{E}[X_{ij}] \\ &\leq \frac{\binom{k}{2}}{n} \\ &\leq \frac{k^2}{2n} \end{aligned}$$

**Lemma (Maximum Load of Entry of Hash Table)**

With probability  $\geq \frac{1}{2}$ , the maximum load when inserting  $k$  elements in a table of size  $n$  using a 2-universal hash family is

$$\leq \sqrt{\frac{2k^2}{n}}$$

When  $k \approx n$ , we expect  $\sqrt{2n}$ .

**Proof.** Let  $C$  be the maximum load entry of the hash table. The number of collisions  $X \geq \binom{C}{2} \sim \frac{C^2}{2}$ . By Markov's inequality,

$$P\left[X \geq \frac{k^2}{n}\right] \leq \frac{1}{2} \implies P\left(X < \frac{k^2}{n}\right) \geq \frac{1}{2}$$

Then,

$$P\left[\frac{C^2}{2} < \frac{k^2}{n}\right] \geq P\left[X < \frac{k^2}{n}\right] \geq \frac{1}{2}$$

**Corollary**

If  $h \in \mathcal{H}$  is a random hash function from a 2-universal family of hash functions, then for any set  $S \subseteq U$  of size  $k \leq \sqrt{n}$ , the probability of  $h$  being perfect for  $S$  is at least  $\frac{1}{2}$ .

## 4.4 Perfect Hashing

Suppose we are given a set of keys  $S$  of size  $n$  in advance. Our goal is to construct a hash table with no collisions and  $O(n)$  memory.

We can do this with a 2-universal hash family, but we would need two layers of hash tables.

For the first layer, use a hash table with  $n$  entries. Let  $\mathcal{H}$  be a 2-universal hash family of maps from  $U \rightarrow [0, n-1]$ . If we pick a random hash function  $h \in \mathcal{H}$ , with probability  $\geq \frac{1}{2}$ , the maximum load of any entry is  $\leq \sqrt{2n}$ .

Since we have the keys, we can check if the maximum load is  $\leq \sqrt{2n}$ , otherwise we can just pick another random hash function and try again. In constantly many tries, with high probability, we will find a hash function  $h$  such that maximum load is  $\leq \sqrt{2n}$ .

Now that we have a first hash table with maximum load  $\sqrt{2n}$ , we can construct a second layer of hash tables. For each entry  $i \in [0, n-1]$ , we will construct a hash table with  $n_i$  entries, where  $n_i$  is the number of keys that hash to  $i$ .

Since  $n_i$  is the load of entry  $i$ , we have  $\frac{n_i(n_i-1)}{2}$  collisions in entry  $i$ . By corollary, we also have that the total number of collisions will be  $\leq n$ , and thus we have

$$\sum_{i=0}^{n-1} n_i^2 = O(n)$$

Let  $\mathcal{H}_i$  be a 2-universal hash family of maps from  $U \rightarrow [0, n_i^2-1]$ . For each entry  $i \in [0, n-1]$ , we can pick a random  $h_i \in \mathcal{H}_i$  with probability  $\geq \frac{1}{2}$ , the maximum load of any entry is  $\leq \sqrt{2}$  (no collisions). Since we have the keys, we can check the maximum load, and if it is not  $\leq \sqrt{2}$ , then try again. In constantly many tries, with high probability, we will find a  $h_i$  which has no collisions.

Thus, we have constructed a hash scheme with no collisions and total memory

$$n + \sum_{i=1}^n n_i^2 = O(n)$$

# Chapter 5

## Graph Sparsification

Often times, graph algorithms for graphs  $G = (V, E)$  have runtimes that depend on the number of edges  $|E|$ .

For example, the runtime of Dijkstra's algorithm is  $O(|E| + |V| \log |V|)$  and the runtime of the Ford-Fulkerson algorithm is  $O(|E| \cdot f^*)$ , where  $f^*$  is the value of the maximum flow.

### Definition: Dense

A graph is dense if

$$|E| = \omega(|V|^{1+\gamma})$$

for  $\gamma \in (0, 1)$ .

### Definition: Sparse

A graph is sparse if

$$|E| = O(|V| \cdot \text{poly log } |V|) = \tilde{O}(|V|)$$

We would like to sparsify a graph, while preserving properties of the graph to reduce to the runtime in practical purposes.

When sparsifying a graph, we may lose some information about the graph, so we will settle with approximately preserving properties.

## 5.1 Minimum Cut

Let  $G = (V, E, w)$  be an undirected graph with nonnegative edge weights  $w_e \geq 0$ . When working with an unweighted graph, then  $w_e = 1$  for all  $e \in E$ . Denote  $n = |V|$  and  $m = |E|$ .



**Definition: Graph Cut**

A cut  $(S, \bar{S})$  is a partition of the vertices  $V$  into two sets  $S$  and  $\bar{S}$ . That is  $V = S \cup \bar{S}$  and  $S \cap \bar{S} = \emptyset$ .

The value of a cut is the sum of the weights of the edges that cross the cut, i.e. edges with one end in  $S$  and the other end in  $\bar{S}$ . Denote  $E(S, \bar{S})$  as the edges that cross the cut, then the value of the cut is

$$w(S, \bar{S}) = \sum_{e \in E(S, \bar{S})} w_e$$

**Definition: Minimum Cut**

A cut of minimum value.

**Definition: Edge Contraction**

Let  $e = uv$  be an edge in  $G = (V, E, w)$ . The contraction of  $e$  is a new graph  $H = (V \cup \{z\} \setminus \{u, v\}, F, w')$ , where  $u, v$  is replaced by  $z$ , any edge  $ux \neq e$  is replaced by  $zx$ , and any edge  $vx \neq e$  is replaced by  $zx$ . Also,  $w'(z, x) = w(u, x)$ .

The contraction of an edge  $e$  is a graph with one less vertex and it may not necessarily be a simple graph. In the case of weighted graphs, we can combine parallel edges into one edge with the sum of the weights.

**Lemma**

Let  $e = uv$  be an edge in  $G = (V, E, w)$  and let  $H$  be obtained by contracting an edge  $e \in E$ .

The value of the minimum cut in  $H$  is at least the value of the minimum cut in  $G$ .

**Algorithm 2** Randomized Minimum Cut

- 1: **Input:** Undirected, unweighted graph  $G = (V, E)$
- 2: **Output:** A minimum cut  $(S, \bar{S})$  of  $G$
- 3: **while**  $n > 2$  **do**
- 4:     Pick  $e = uv$  uniformly at random from  $E$
- 5:     Contract  $e$
- 6: **if**  $n = 2$  **then return**  $(S, \bar{S})$  induced by the two vertices in  $V$

**Theorem (Karger)**

The randomized minimum cut algorithm outputs a minimum cut with probability at least  $\frac{2}{n(n-1)}$ .

**Proof.** Let  $(S, \bar{S})$  be a minimum cut of  $G$  and let  $c = w(S, \bar{S})$ . If we never contract an edge from  $E(S, \bar{S})$ , then the algorithm succeeds, as we will output  $w(S, \bar{S})$ .

Computing the probability that an edge from  $E(S, \bar{S})$  is contracted in one iteration of the algorithm. Each vertex is a cut, so each vertex has degree at least  $c$ . Hence, we know that at least  $(n - i + 1) \cdot \frac{c}{2}$  edges remain.

The probability that we contract an edge from  $E(S, \bar{S})$  is

$$\frac{w(S, \bar{S})}{\# \text{ edges}} \leq \frac{c}{(n - i + 1) \cdot \frac{c}{2}} = \frac{2}{n - i + 1}$$

Hence, the probability that we never contract an edge from  $E(S, \bar{S})$  is at least

$$\prod_{i=1}^{n-2} \left(1 - \frac{2}{n - i + 1}\right) = \prod_{i=3}^n \left(1 - \frac{2}{i}\right) = \frac{2}{n(n-1)} = \frac{1}{\binom{n}{2}}$$

To improve the above probability, we can run the algorithm multiple times and output the minimum cut over all iterations. If we repeat  $t$  times, then the failure probability is at most

$$\left(1 - \frac{2}{n(n-1)}\right)^t$$

If we set  $t = 2n(n-1)$ , then we get a failure probability of at most

$$\left(1 - \frac{2}{n(n-1)}\right)^{2n(n-1)} = \left(\left(1 - \frac{2}{n(n-1)}\right)^{n(n-1)/2}\right)^4 \sim \frac{1}{e^4}$$

Runtime: Each iteration of the algorithm takes  $O(m)$  time, and we run the algorithm  $O(n^2)$  times, so the total runtime is  $O(n^2m)$ .

### Corollary (Karger)

There are at most  $\binom{n}{2}$  minimum cuts in a graph.

**Proof.** Each minimum cut is preserved with probability at least  $\frac{1}{\binom{n}{2}}$ . Since the events that each minimum cut is preserved are disjoint (and sum of probability is  $\leq 1$ ), there can be at most  $\binom{n}{2}$  minimum cuts.

### Lemma

If  $c$  is the minimum cut in  $G$ , then there are at most  $n^{2\alpha}$  cuts of value  $k \leq \alpha c$  in  $G$ .

## 5.2 Graph Sparsification Algorithm

We need to set  $p$  to be the corrected expected value for both the number of edges in  $H$  and the value of each cut in  $H$ . After that, we need to prove concentration bounds for values of all cuts in  $H$ , simultaneously.

---

**Algorithm 3** Randomized Sparsification

---

- 1: **Input:** Undirected, unweighted graph  $G = (V, E)$  and parameter  $\varepsilon > 0$   
2: **Output:** A sparse weighted graph  $H = (V, F, w)$  such that for every cut  $(S, \bar{S})$ , we have

$$(1 - \varepsilon) \cdot w(S, \bar{S}) \leq w_H(S, \bar{S}) \leq (1 + \varepsilon) \cdot w(S, \bar{S})$$

- 3:  $p \in (0, 1)$  be a parameter  
4: For each edge  $e \in E$ , include  $e$  in  $F$  with probability  $p$ , and if included, set  $w_H(e) = \frac{1}{p}$
- 

We can do this using Chernoff-Hoeffding bounds, then show that there are not too many small cuts in  $G$ , and thus, the probability that we have a bad cut in  $H$  is small. We can then use union bound to prove all cuts are concentrated.

**Theorem (Karger)**

Let  $c$  be the value of the minimum cut in  $G$ . Set

$$p = \frac{15 \log n}{\varepsilon^2 c}$$

With probability  $\geq 1 - \frac{4}{n}$ , the above algorithm outputs a graph  $H = (V, F, w_H)$  with  $|F| = O(p \cdot |E|)$  such that for every cut  $(S, \bar{S})$  in  $G$ :

$$(1 - \varepsilon) \cdot w(S, \bar{S}) \leq w_H(S, \bar{S}) \leq (1 + \varepsilon) \cdot w(S, \bar{S})$$

**Proof.** Let  $H = (V, F, w_H)$  be the graph output by the algorithm. Take a cut  $(S, \bar{S})$ . Denote  $k = w(S, \bar{S})$ . Let  $X_e$  be the indicator random variable for the event that  $e \in F$ .

Then,  $w_H(S, \bar{S}) = \sum_{e \in E(S, \bar{S})} \frac{X_e}{p}$  and  $|F| = \sum_{e \in E} X_e$ .

Hence, the expected values are

- $\mathbb{E}[|F|] = \sum_{e \in E} \mathbb{E}[X_e] = p \cdot m$
- $\mathbb{E}[w_H(S, \bar{S})] = \sum_{e \in E(S, \bar{S})} \mathbb{E}[w_H(e)] = \sum_{e \in E(S, \bar{S})} \mathbb{E}[X_e/p] = k$

Now compute concentration bounds for  $|F|$  and  $w_H(S, \bar{S})$ :

- For  $|F|$ ,

$$P[|F| \geq (1 + \varepsilon) \cdot p \cdot m] \leq e^{-\varepsilon^2 \cdot \mathbb{E}[|F|]/3} = e^{-\varepsilon^2 \cdot p \cdot m/3} \leq \frac{1}{n^2}$$

- For  $w_H(S, \bar{S})$ , note that  $p \cdot w_H(S, \bar{S})$  is a sum of independent random variables with values in  $0, 1$ . We can use Chernoff bounds to get

$$P[|w_H(S, \bar{S}) - k| \geq \varepsilon \cdot k] \leq 2 \cdot e^{-\varepsilon^2 \cdot kp/3} = 2n^{-5k/c}$$

Note that  $k \geq c$ , as  $c$  is the value of the minimum cut of  $G$ . The above is the probability that a single cut deviates from its expectation. We want all cuts. Using union bound,

$$\begin{aligned}
P[\text{a cut deviates from expectation}] &\leq \sum_{S \subseteq V} P[|w_H(S, \bar{S}) - k| \geq \varepsilon \cdot k] \\
&\leq \sum_{\alpha=1,2,4,8,\dots} \sum_{\substack{S \subseteq V \\ \alpha c \leq |w_G(S, \bar{S})| \leq 2 \cdot \alpha c}} P[|w_H(S, \bar{S}) - k| \geq \varepsilon \cdot k] \\
&\leq \sum_{\alpha=1,2,4,8,\dots} n^{4\alpha} \cdot P[|w_H(S, \bar{S}) - k| \geq \varepsilon \cdot k | \alpha c \leq k \leq 2\alpha c] \\
&\leq \sum_{\alpha=1,2,4,8,\dots} n^{4\alpha} \cdot 2 \cdot n^{-5\alpha} \\
&= \sum_{\alpha=1,2,4,8,\dots} n^{-\alpha} \leq \frac{4}{n}
\end{aligned}$$

We have assumed that the graph has a large min-cut value:  $c = \Omega(\log n)$ . This assumption is so that  $p < 1$ . To remove the assumption, we can use non-uniform sampling of edges. If we choose this non-uniform sampling carefully, we can get a sparse graph which approximates all cuts with high probability.

**Definition: Strong Connectivity**

A  $k$ -strong component is a maximal induced subgraph that is  $k$ -edge-connected.

# Chapter 6

## Algebraic Techniques: Fingerprinting, Polynomial Identity Testing, and Parallel Matching Algorithms

### 6.1 Verifying String Equality

Suppose Alice and Bob each maintain the same large database of information. They want to check if their databases are consistent.

However, transmission of all data is expensive and sending the entire database is not feasible. Say Alice's database is given by bits  $(a_1, \dots, a_n)$  and Bob's database is given by bits  $(b_1, \dots, b_n)$ . A deterministic consistency check requires both of them to communicate  $n$  bits.

Problem
Given strings $(a_1, \dots, a_n)$ and $(b_1, \dots, b_n)$ , check if they are equal.

Fingerprinting: Let  $a = \sum_{i=1}^n a_i 2^{i-1}$  and  $b = \sum_{i=1}^n b_i 2^{i-1}$ . Let  $F_p(x) = x \bmod p$  be a fingerprinting function for a prime  $p$ .

**Protocol:**

1. Alice picks a random prime  $p$  and sends  $(p, F_p(a))$  to Bob.
2. Bob checks whether  $F_p(a) \equiv F_p(b) \bmod p$  and sends

$$\begin{cases} 1 & \text{if values are equal} \\ 0 & \text{otherwise} \end{cases}$$

The total bits communicated is  $O(\log p)$  bits, dominated by Alice's message. If the two strings are equal, then the protocol is always right.

If  $(a_1, \dots, a_n) \neq (b_1, \dots, b_n)$ , then  $a \neq b$ . If a number  $M$  is in  $\{-2^n, \dots, 2^n\}$ , then the number of distinct primes  $p|M$  is  $< n$ .

- Each prime divisor of  $M$  is  $\geq 2$ , so if  $M$  has  $t$  distinct prime divisors, then  $|M| > 2^t$ .
- $|M| \leq 2^n \implies t \leq n$

Thus,  $F_p(a) \equiv F_p(b)$  if and only if  $p|a - b$ . The protocol fails for at most  $n$  choices of  $p$ .

### Theorem (Prime Number Theorem)

There are  $\frac{m}{\log m}$  primes among the first  $m$  positive integers.

Choosing  $p$  among the first  $tn \log(tn)$  integers, we have

$$P[F_p(a) \equiv F_p(b)] \leq \frac{n}{tn \log(tn) / \log(tn \log(tn))} = \tilde{O}\left(\frac{1}{t}\right)$$

The number of bits sent is  $\tilde{O}(\log t + \log n)$ . Choosing  $t = n$  solves it.

## 6.2 Polynomial Identity Testing

### Problem

**Input:** two polynomials  $P(x), Q(x)$ .

**Output:** are they equal?

Two polynomials are equal if and only if all their coefficients are equal.

We cannot just compare coefficients since polynomials may sometimes be given implicitly. For example, we may want to test if  $P_1(x) \cdot P_2(x) = P_3(x)$ . If  $P_1, P_2$  have degree  $\leq n$ , then  $\deg(P_3) \leq 2n$ , otherwise the problem is trivial.

Multiplication of two polynomials of degree  $n$  take  $O(n \log n)$  by arithmetic operation of fast Fourier transform. Polynomial evaluation also takes  $O(n)$  arithmetic operations, if we are given the coefficients.

### Lemma (Roots of Univariate Polynomials)

Let  $\mathbb{F}$  be a field and  $P(x) \in \mathbb{F}[x]$  be a nonzero univariate polynomial of degree  $d$ . Then,  $P(x)$  has at most  $d$  roots in  $\mathbb{F}$ .

$P_1, P_2 = P_3 \iff P_1 P_2 - P_3 = 0$ , so there are at most  $2n$  roots by lemma. Take  $S \subseteq \mathbb{F}$  of size  $4n$ . Let  $a \in S$  be random. Compute  $Q(a) = P_3(a) - P_1(a)P_2(a)$ .

$$P_{a \in S}[Q(a) = 0] \leq \frac{\deg(Q)}{|S|} \leq \frac{2n}{4n} = \frac{1}{2}$$

**Lemma (Ore-Schwartz-Zippel-de Millo-Lipton)**

Let  $\mathbb{F}$  be a field and  $P(x_1, \dots, x_n) \in \mathbb{F}[x_1, \dots, x_n]$  be a nonzero polynomial of degree  $\leq d$ . Then, for any set  $S \subseteq \mathbb{F}$ , we have

$$\Pr[P(a_1, \dots, a_n) = 0 | a_i \in S] \leq \frac{d}{|S|}$$

## 6.3 Bipartite Matching

**Problem**

**Input:** bipartite graph  $G = (L, R, E)$  with  $|L| = |R| = n$ .

**Output:** does  $G$  have a perfect matching?

A perfect matching corresponds to a permutation  $S_n$ .

Let  $X \in \mathbb{F}^{n \times n}$  be such that

$$X_{i,j} = \begin{cases} y_{i,j} & \text{if there is an edge between } (i, j) \in L \times R \\ 0 & \text{otherwise} \end{cases}$$

The determinant is

$$\det(X) = \sum_{\sigma \in S_n} (-1)^\sigma \prod_{i=1}^n X_{i,\sigma(i)}$$

$G$  has a perfect matching if and only if  $\det(X)$  is a nonzero polynomial. Therefore, testing if  $G$  has a perfect matching is a special case of Polynomial Identity Testing.

**Algorithm:** Evaluate  $\det(X)$  at a random value for the variables  $y_{i,j}$ .

## 6.4 General Matching

**Problem**

**Input:** undirected graph  $G = (V, E)$  where  $|V| = 2n$ .

**Output:** does  $G$  have a perfect matching?

**Definition: Tutte Matrix**

$T_G$  is defined by the  $2n \times 2n$  matrix: let  $F$  be an arbitrary orientation of the edges in  $E$ . Then,

$$[T_G]_{i,j} = \begin{cases} x_{i,j} & \text{if } (i,j) \in F \\ -x_{i,j} & \text{if } (j,i) \in F \\ 0 & \text{otherwise} \end{cases}$$

**Theorem (Tutte 1947)**

$G$  has a perfect matching if and only if  $\det(T_G) \neq 0$ .

## 6.5 Parallel Algorithms and Isolation Lemma

Often times in parallel computation, when solving a problem with many possible solutions, we want all processors working towards the same solution. We need to isolate a specific solution without knowing any element of the solution space.

The solution is to implicitly choose a random order on the feasible solutions and require processors to find solution of lowest rank in this order. We can use this to compute the minimum weight perfect matching.

**Lemma (Isolation Lemma)**

Given a set system over  $[n] := \{1, \dots, n\}$ , if we assign a random weight function  $w : [n] \rightarrow [2n]$ , then the probability that there is a unique solution minimum weight set is at least  $\frac{1}{2}$ .

This could be quite counter-intuitive. A set system can have  $\Omega(2^n)$  sets. On average, there are  $\Omega\left(\frac{2^n}{2n^2}\right)$  sets of a given weight, as max weight is  $\leq 2n^2$ . Isolation lemma tells us that with high probability, there is only one set of minimum weight.

**Proof.** Let  $\mathcal{S}$  be our set system and  $v \in [n]$ . Let  $\mathcal{S}_v$  be a family of sets from  $\mathcal{S}$  which contain  $v$ , and  $\mathcal{N}_v$  be the family of sets from  $\mathcal{S}$  which do not contain  $v$ . Let

$$\alpha_v := \min_{A \in \mathcal{N}_v} w(A) - \min_{B \in \mathcal{S}_v} w(B \setminus \{v\})$$

- $\alpha_v < w(v)$  implies  $v$  does not belong to any minimum weight set.
- $\alpha_v > w(v)$  implies  $v$  belongs to every minimum weight set.
- $\alpha_v = w(v)$  implies  $v$  is ambiguous.



$\alpha_v$  is independent of  $w(v)$  and  $w(v)$  is chosen uniformly at random from  $[2n]$ .

$$P[v \text{ ambiguous}] \leq \frac{1}{2n} \implies_{\text{union bound}} P[\exists \text{ ambiguous element}] \leq \frac{1}{2}$$

If two different sets  $A, B$  have minimum weight, then any element in  $A \Delta B$  must be ambiguous. The probability this happens  $\leq \frac{1}{2}$ .

# Chapter 7

## Sublinear Time Algorithms

We cannot answer for all, there exists, or exactly type statements. We can answer for most, averages, or approximate type statements.

If  $N$  is the input size, then a sublinear time algorithm may not read in all of  $N$ , so we want an algorithm in  $o(N)$ .

### 7.1 Approximate Diameter

#### Definition: $\alpha$ -Multiplicative Approximation

$$\frac{1}{\alpha}X_{\max} \leq X \leq \alpha X_{\max}$$

#### Approximate Diameter of a Point Set

**Input:**  $m$  points and a distance matrix  $D$  such that

- $D_{ij}$  is the distance from  $i$  to  $j$ .
- $D$  is symmetric and satisfies triangle inequality;  $D_{ij} \leq D_{ik} + D_{kj}$ .

The input is given in adjacency matrix representation. Input size is  $N = m^2$ .

The diameter  $D_{ab}$  is the maximal distance between indices  $a$  and  $b$ .

**Output:** Indices  $k, l$  such that

$$D_{kl} \geq \frac{D_{ab}}{2}$$

This will be a 2-approximation algorithm.

**Algorithm:** Pick  $k$  arbitrary, pick  $l$  to maximize  $D_{kl}$ , output indices  $k, l$ .

Correctness:

$$D_{ab} \leq D_{ak} + D_{kb} \leq D_{kl} + D_{kl} = 2D_{kl}$$

Runtime:  $O(m) = O(N^{1/2})$

### 7.1.1 Lower Bound

Let  $D$  be the distance matrix where  $D_{i,i} = 0$  for all  $i \in [m]$  and  $D_{i,j} = 1$  otherwise. Let  $D'$  be the same matrix as  $D$  except for one pair  $(a, b)$ , we make

$$D'_{ab} = D'_{ba} = 2 - \delta$$

Check that  $D'$  satisfies properties of a distance matrix. We can prove that it would take  $\Omega(N)$  time (number of queries) to decide if diameter is 1 or  $2 - \delta$ .

## 7.2 Connected Components

### Approximate Number of Connected Components

**Input:** graph  $G = (V, E)$  in adjacency list, precision parameter  $\varepsilon > 0$ .

**Output:** if  $C$  is number of connected components of  $G$ , output with probability  $\geq \frac{3}{4}$ ,  $C'$  such that

$$|C' - C| \leq \varepsilon n$$

### Lemma (Number of Connected Components)

Let  $G = (V, E)$  be a graph. For vertex  $v \in V$ , let  $n_v$  be the number of vertices in connected component of  $v$ . Let  $C$  be the number of connected components of  $G$ . Then,

$$C = \sum_{v \in V} \frac{1}{n_v}$$

Take a sample of vertices from  $G$ , compute  $n_v$  and output the normalization. The problem is that computing  $n_v$  make take linear time. If  $n_v$  is large, then  $\frac{1}{n_v}$  is small, so we can drop it.

### Lemma (Estimating Number of Connected Components)

Let  $n'_v = \min\{n_v, \varepsilon\}$ , then

$$\left| \sum_{v \in V} \frac{1}{n_v} - \sum_{v \in V} \frac{1}{n'_v} \right| \leq \frac{\varepsilon n}{2}$$

**Proof.** Let  $U = \{v \in V | n_v > 2/\varepsilon\}$ .

$$\begin{aligned} \left| \sum_{v \in U} \left( \frac{\varepsilon}{2} - \frac{1}{n_v} \right) \right| &= \sum_{v \in U} \left( \frac{\varepsilon}{2} - \frac{1}{n_v} \right) \\ &\leq \frac{|U| \varepsilon}{2} \\ &\leq \frac{n\varepsilon}{2} \end{aligned}$$

Sample vertex  $v$ , run BFS, and stop if we see  $2/\varepsilon$  vertices.

**Algorithm:** Choose  $s = \Theta(1/\varepsilon^2)$  vertices  $v_1, \dots, v_s$  uniformly at random. Compute  $n'_{v_i}$  using BFS. Return

$$C' = \frac{n}{s} \sum_{i=1}^s \frac{1}{n'_{v_i}}$$

Runtime:  $\Theta(1/\varepsilon^2)$  vertices sampled, each run takes  $O(1/\varepsilon^2)$  time to compute. Adding results takes  $O(s) = O(1/\varepsilon^2)$  time. The total runtime is  $O(1/\varepsilon^4)$ .

Correctness: To prove correctness, we show that with probability  $\geq \frac{3}{4}$ , we have

$$\left| \frac{n}{s} \sum_{i=1}^s \frac{1}{n'_{v_i}} - \sum_{v \in V} \frac{1}{n_v} \right| \leq \varepsilon n$$

Dividing both sides by  $\frac{n}{s}$ ,

$$\left| \sum_{i=1}^s \frac{1}{n'_{v_i}} - \frac{s}{n} \sum_{v \in V} \frac{1}{n_v} \right| \leq \varepsilon s$$

By previous lemma and triangle inequality, it is enough to prove that with high probability  $\geq \frac{3}{4}$

$$\left| \sum_{i=1}^s \frac{1}{n'_{v_i}} - \frac{s}{n} \sum_{v \in V} \frac{1}{n_v} \right| \leq \frac{\varepsilon s}{2}$$

For the  $i$ th estimate, have a random variable  $X_i = \frac{1}{n'_v}$  with probability  $\frac{1}{n}$ ,  $a_i = 0$ , and  $b_i = 1$ .

We can use Hoeffding's inequality. So  $X = \sum_{i=1}^s X_i$ .

$$\mathbb{E}[X] = \sum_{i=1}^s \mathbb{E}[X_i] = \sum_{i=1}^s \sum_{j=1}^n \frac{1}{n'_{v_j}} \cdot \frac{1}{n} = \frac{s}{n} \sum_{j=1}^n \frac{1}{n'_{v_j}}$$

This is exactly the quantity in the absolute value. So pick  $l = \varepsilon \cdot s/2$

$$P[\text{fail}] \leq 2 \exp \left( -\frac{2\varepsilon^2 s^2}{4s} \right) = 2 \exp \left( -\frac{\varepsilon^2 s}{2} \right) \leq \frac{1}{4}$$

# Chapter 8

## Random Walks and Markov Chains

### 8.1 Random Walks

**Definition: Random Walk**

Given a graph  $G = (V, E)$ , a random walk starts from a vertex  $v_0$  and at each time step, it moves uniformly to a random neighbour of the current vertex in  $G$ .

$$v_{t+1} \leftarrow_R N_G(v_t)$$

Basic questions for random walks:

- Stationary distribution: does the random walk converge to a stable distribution?
- Mixing time: how long does it take for the walk to converge to the stationary distribution?
- Mean hitting time: starting from  $v_0$ , what is the expected number of steps until it reaches another vertex  $v_f$ ?
- Cover time: how long does it take to reach every vertex of the graph at least once?

**Example:** Suppose  $G = K_n$  and  $a, b \in V$ . We can ask all the questions above.

- Mean hitting time: Let  $X$  be the number of steps for random walk from  $a \rightarrow b$ .

$$\begin{aligned} P[X = 1] &= \frac{1}{n-1} \\ P[X = 2] &= \frac{n-2}{n-1} \cdot \frac{1}{n-1} \\ P[X = k] &= \left(\frac{n-2}{n-1}\right)^{k-1} \cdot \frac{1}{n-1} \end{aligned}$$

$X$  is a geometric random variable with  $p = \frac{1}{n-1}$

$$\mathbb{E}[X] = \frac{1}{p} = n - 1$$

- Stationary distribution: If at time  $t$ , we have  $\left(\frac{1}{n}, \dots, \frac{1}{n}\right)$ , then at  $t + 1$ , the probability that we come back to any vertex is  $(n - 1) \cdot \frac{1}{n-1} \cdot \frac{1}{n} = \frac{1}{n}$ .
- Cover time: is similar to the coupon collector problem.

## 8.2 Markov Chains

A random walk is a special kind of stochastic process:

$$P[X_t = v_t | X_0 = v_0, \dots, X_{t-1} = v_{t-1}] = P[X_t = v_t | X_{t-1} = v_{t-1}]$$

in that the probability that we are at  $v_t$  at time  $t$  depends only on the state of our process at time  $t - 1$ .

### Definition: Markov Chain

A process in which the probability of an event only depends on the previous event.

A Markov chain can be seen as weighted directed graph, where the vertex is a state of the Markov chain and an edge  $(i, j)$  corresponds to the transition probability from  $i$  to  $j$ .

### Definition: Irreducible

A Markov chain is irreducible if the underlying directed graph is strongly connected.

### Definition: Transition Matrix

The weighted adjacency matrix  $P \in \mathbb{R}^{n \times n}$  of the weighted directed graph of a Markov chain.

$P_{i,j}$  corresponds to the transition probability from  $i$  to  $j$ .

### Definition: Probability Vector

$p_t \in \mathbb{R}^n$  is the probability vector where  $p_t(i) := P[\text{state } i \text{ at time } t]$ .

### Definition: Transition

$$p_{t+1} = p_t \cdot P$$

**Definition: Period**

The period of a state  $i$  is

$$\gcd\{t \in \mathbb{N} : P_{i,i}^t > 0\}$$

That is, the GCD of all times  $t$  such that the probability of starting at  $i$  and begin back at  $i$  at time  $t$  is positive.

The period of the square graph on 4 vertices is 2, since for any vertex, we have to take an even number of steps to get back to that vertex.

**Definition: Aperiodic**

State  $i$  is aperiodic if its period is equal to 1.

A Markov chain is aperiodic if all states are aperiodic, otherwise it is periodic.

Bipartite graphs yield periodic Markov chains.

**Lemma**

For any finite, irreducible, and aperiodic Markov chain, there exists  $T < \infty$  such that

$$P_{i,j}^t > 0 \text{ for any } i, j \in V \text{ and } t \geq T$$

## 8.3 Stationary Distributions

**Definition: Stationary Distribution**

A stationary distribution of a Markov chain is a probability distribution  $\pi \in \mathbb{R}^n$  such that

$$\pi P = \pi$$

Informally,  $\pi$  is an equilibrium/fixed point state, as we have  $\pi = \pi P^t$  for any  $t \geq 0$ .

The intuition is if we run a finite, irreducible, and aperiodic Markov chain long enough, we will converge to a unique stationary distribution.

**Definition: Total Variational Distance**

Given two distributions  $p, q \in \mathbb{R}^n$ ,

$$\Delta_{TV}(p, q) = \frac{1}{2} \sum_{i=1}^n |p_i - q_i| = \frac{1}{2} \|p - q\|_1$$

**Proposition**

$p_t$  converges to  $q$  if and only if  $\lim_{t \rightarrow \infty} \Delta_{TV}(p_t, q) = 0$ .

**8.3.1 Mixing Time****Definition: Mixing Time**

The  $\varepsilon$ -mixing time of a Markov chain is the smallest  $t$  such that

$$\Delta_{TV}(p_t, \pi) \leq \varepsilon$$

regardless of the initial starting distribution  $p_0$ .

For the complete graph  $K_n$ , we have 0s along the diagonal of  $P$  and  $\frac{1}{n-1}$  everywhere else. Let  $J$  be the all one's matrix and  $I$  be the identity matrix,

$$P = \frac{1}{n-1}J - \frac{1}{n-1}I = \frac{1}{n-1}\mathbf{1}^T\mathbf{1} - \frac{1}{n-1}I$$

The eigenvalue  $\lambda_1 = 1$  gives  $1P = \mathbf{1}$ . The characteristic polynomial of  $P$  is

$$P = \lambda_1 v_1 v_1^T + \lambda_2 v_2 v_2^T + \cdots + \lambda_n v_n v_n^T$$

$$P^t = \lambda_1^t v_1 v_1^T + \lambda_2^t v_2 v_2^T + \cdots + \lambda_n^t v_n v_n^T$$

$\lambda_2 = \cdots = \lambda_n = -\frac{1}{n-1}$  and the corresponding  $v_1, \dots, v_n$  are orthonormal.

**Definition: Spectral Gap**

The spectral gap  $\lambda$  is given by

$$\lambda = \min\{1 - \alpha_2, 1 - |\alpha_n|\}$$

so that  $|\alpha_i| \leq 1 - \lambda$  for  $2 \leq i \leq n$ .

**Theorem**

The  $\varepsilon$ -mixing time of a random walk is upper bounded by

$$\frac{1}{\lambda} \log \left( \frac{n}{\varepsilon} \right)$$



## 8.4 Hitting Time

### Definition: Hitting Time

Given states  $i, j$  in a Markov chain,

$$T_{i,j} := \min\{t \geq 1 : X_t = j, X_0 = i\}$$

$T_{i,j} = \infty$  if the Markov chain never visits  $j$  starting from  $i$ .

### Definition: Mean Hitting Time

$$\tau_{i,j} := \mathbb{E}[T_{i,j}]$$

### Lemma (Hitting Time Lemma)

For any finite, irreducible, and aperiodic Markov chain and for any two states  $i, j$ , not necessarily distinct, we have

$$P[T_{i,j} < \infty] = 1 \text{ and } \mathbb{E}[T_{i,j}] < \infty$$

**Proof.** We know we can find  $M < \infty$  such that  $(P^M)_{i,j} > 0$  for all  $i, j$ .

## 8.5 Linear Algebra Background

### Definition: Eigenvalue/Eigenvector

Given a square matrix  $A \in \mathbb{R}^{n \times n}$ ,  $\lambda \in \mathbb{C}$  is an eigenvalue of  $A$  if there is a vector  $v \in \mathbb{C}^n$  such that  $Av = \lambda v$ .

$v$  is the eigenvector corresponding to  $\lambda$ .

### Definition: Spectral Radius

Denoted  $\rho(A)$ , is the maximum absolute value of the eigenvalues of  $A$ .

### Theorem (Gelfand's Formula)

$$\rho(A) = \lim_{t \rightarrow \infty} \|A^t\|_F^{1/t}$$

where  $\|A\|_F = \text{tr}(A^T A)^{1/2}$ .

**Definition: Geometric Multiplicity**

An eigenvalue  $\lambda$  has geometric multiplicity  $k$  if the space of eigenvectors of  $A$  with eigenvalue  $\lambda$  has dimension  $k$ .

That is, if the dimension of the nullspace of  $A - \lambda I$  is  $k$ .

**Definition: Algebraic Multiplicity**

An eigenvalue  $\lambda$  of  $A$  has algebraic multiplicity  $k$  if  $(t - \lambda)^k$  is the highest power of  $t - \lambda$  dividing  $\det(tI - A)$ .

**Lemma (Positivity Lemma)**

If  $A \in \mathbb{R}^{n \times n}$  is a positive matrix and  $u, v \in \mathbb{R}^n$  are distinct vectors such that  $u \geq v$ , then  $Au > Av$ .

Moreover, there exists  $\varepsilon > 0$  such that  $Au > (1 + \varepsilon)Av$ .

**Proof.**

$$[A(u - v)]_i = \sum_j A_{ij}(u - v)_j \geq \min_{i,j} A_{ij} \sum_{j=1}^n (u - v)_j$$

Since  $u_j \geq v_j$ , there is one index  $k$  such that  $u_k > v_k$ , we have

$$\sum_j (u - v)_j \geq u_k - v_k > 0$$

**8.5.1 Perron-Frobenius****Theorem (Perron)**

Let  $A \in \mathbb{R}^{n \times n}$  be a positive matrix. Then, the following hold:

- $\rho(A)$  is an eigenvalue, and it has a positive eigenvector.
- $\rho(A)$  is the only eigenvalue in the complex circle  $|\lambda| = \rho(A)$ .
- $\rho(A)$  has geometric multiplicity 1.
- $\rho(A)$  has algebraic multiplicity 1.

**Proof.** (first point) By definition of  $\rho(A)$ , there is an eigenvalue  $\lambda \in \mathbb{C}$  such that  $|\lambda| = \rho(A)$ . Let  $v$  be the corresponding eigenvector. Let  $u$  be the vector defined by  $u_i = |v_i|$ . Then we have

$$(Au)_i = \sum_j A_{ij}u_j \geq \left| \sum_j A_{ij}v_j \right| = |\lambda v_i| = \rho(A) \cdot u_i$$

so  $Au \geq \rho(A)u$ .

If the inequality is strict, then we have

$$A^2u > \rho(A) \cdot Au$$

and there is some positive  $\varepsilon > 0$  such that

$$A^2u \geq (1 + \varepsilon)\rho(A)Au$$

By induction, we have

$$A^n u \geq (1 + \varepsilon)^{n-1} \cdot \rho(A)^{n-1} \cdot Au$$

By Gelfand's formula, we would have

$$\rho(A) = \lim_{n \rightarrow \infty} \|A^n\|_f^{1/n} \geq (1 + \varepsilon)\rho(A)$$

which is a contradiction. So equality must hold.

(second point) We just proved that  $\rho(A)$  is an eigenvalue, with eigenvector  $u \geq 0$ . Note that  $u > 0$  since  $\rho(A)u_i = (Au)_i > 0$ . The only eigenvalue on the complex circle  $|\mu| = \rho(A)$  is  $\rho(A)$ . If we had another eigenvalue  $\lambda \neq \rho(A)$  in the circle  $|\mu| = \rho(A)$ , where  $z$  is the eigenvalue corresponding to  $\lambda$ , then we know that  $w$  defined as  $w_i = |z_i|$  satisfies

$$Aw = \rho(A)w \Leftrightarrow \sum_j A_{ij}w_j = \rho(A) \cdot |z_i| = |\lambda z_i| = \left| \sum_j A_{ij}z_j \right|$$

for every  $1 \leq i \leq n$ .

**Lemma**

If the conditions above hold, then there is  $\alpha \in \mathbb{C}$  nonzero such that  $\alpha z \geq 0$ .

But if  $\alpha z \geq 0$  and a nonzero vector, we have

$$\lambda(\alpha z) = \alpha(\lambda z) = \alpha(Az) = A(\alpha z) \geq 0$$

since  $A$  is positive and  $\alpha z \geq 0$ . Thus, we know that  $\lambda$  is a nonnegative number. However,  $\rho(A)$  is the only nonnegative number in the circle  $|\mu| = \rho(A)$ .

(third point) Suppose not. Let  $u, v$  be two linearly independent eigenvectors for  $\rho(A)$ . We can assume that both  $u, v$  are real vectors. If  $u$  was complex, then  $u = \varphi + i\psi$ , but  $\rho(A)u = Au = A\varphi + iA\psi = \rho(A)\varphi + i\rho(A)\psi$ , so we can just choose  $u = \varphi$ .

Let  $\beta > 0$  such that  $u - \beta v \geq 0$  and at least one entry is 0.  $u - \beta v \neq 0$  since the vectors are linearly independent. But for each  $1 \leq i \leq n$ ,

$$\rho(A) \cdot (u - \beta v)_i = (A(u - \beta v))_i > 0$$

which contradicts our choice of  $\beta$ . Thus, there cannot be two linearly independent vectors.

**Theorem (Perron-Frobenius)**

If a nonnegative matrix  $A \in \mathbb{R}^{n \times n}$  is aperiodic and irreducible, then the following hold:

- $\rho(A)$  is an eigenvalue and it has positive eigenvector.
- $\rho(A)$  is the only eigenvalue in the complex circle  $|\lambda| = \rho(A)$ .
- $\rho(A)$  has geometric multiplicity 1.
- $\rho(A)$  has algebraic multiplicity 1.

**Proof.**  $A$  being aperiodic and irreducible implies that there is  $m > 0$  such that  $A^m$  has all positive entries. Apply Perron's theorem to  $A^m$  and note that the eigenvalues of  $A^m$  are  $\lambda_i^m$ , where  $\lambda_i$  are the eigenvalues of  $A$ .

## 8.6 Fundamental Theorem of Markov Chains

**Definition: Return Time**

For state  $i$

$$T_{i,i} := \min\{t \geq 1 : X_t = i, X_0 = i\}$$

**Definition: Expected Return Time**

$$\tau_{i,i} := \mathbb{E}[T_{i,i}]$$

The return time from state  $i$  to itself is  $T_{i,i}$  and the expected return time is  $\tau_{i,i} := \mathbb{E}[T_{i,i}]$ .

**Theorem (Fundamental Theorem of Markov Chains)**

Any finite, irreducible, and aperiodic Markov chain has the following properties:

1. There exists a unique stationary distribution  $\pi$ , where  $\pi_i > 0$  for all  $i \in [n]$ .
2. The sequence of distributions  $\{p_t\}_{t \geq 0}$  will converge to  $\pi$ , no matter what the initial distribution is, i.e. for every distribution  $p_0 \in \mathbb{R}_{\geq 0}^n$ ,

$$\lim_{t \rightarrow \infty} p_0 P^t = \pi$$

3.

$$\pi_i = \lim_{t \rightarrow \infty} P_{i,i}^t = \frac{1}{\tau_{i,i}}$$

**Proof.** If our underlying graph is undirected, it is easy to guess the stationary distribution:

$$\pi_i = \frac{d_i}{2m}, m = |E|$$

If  $A_G$  is the adjacency matrix of  $G$  and  $D = \text{diag}(d_1, \dots, d_n)$  is the transition matrix, then

$$P = D^{-1}A_G$$

- $P$  is not symmetric, but it is similar to a symmetric matrix

$$D^{1/2}PD^{-1/2} = D^{1/2}D^{-1}A_GD^{-1/2} = D^{-1/2}A_GD^{-1/2} = P'$$

- $P$  and  $P'$  have the same eigenvalues and  $P'$  has only real eigenvalues.
- Eigenvectors of  $P$  are  $D^{-1/2}v_i$  where  $v_i$  are eigenvectors of  $P'$ .  
Moreover,  $v_i$  can be taken to form an orthonormal basis.
- If a graph is strongly connected, then Perron-Frobenius for irreducible nonnegative matrices.
  - Unique eigenvector whose eigenvalue has maximum absolute value.
  - Eigenvector has all positive coordinates.
  - Eigenvalue is positive.
- This eigenvector is  $\pi$ , so all random walks converge to  $\pi$ .

## 8.7 Page Rank

### Definition: Page Rank

A directed graph describing relationships between set of webpages. There is a directed edge  $(i, j)$  if there is a link from page  $i$  to page  $j$ .

Goal: rank how important a page is.

### Page Rank Algorithm

1. Each page has pagerank value  $\frac{1}{n}$
2. In each step, each page:
  1. Divides its pagerank value equally to its outgoing link
  2. Sends these equal shares to the pages it points to
  3. Updates its new pagerank value to be the sum of shares it receives

Equilibrium of pagerank values equal to probabilities of stationary distribution of random walk

$$P \in \mathbb{R}^{n \times n}, P_{i,j} = \frac{1}{\delta^{out}(i)}$$

Pagerank values and transition probabilities satisfy same equations:

$$p_{t+1}(j) = \sum_{i:(i,j) \in E} \frac{p_t(i)}{\delta^{out}(i)} \implies p_{t+1} = p_t \cdot P$$

If the graph is finite, irreducible, and aperiodic, then the fundamental theorem guarantees stationary distribution.

In practice, the graph may not satisfy the fundamental theorem's conditions. We can modify the original graph as follows:

- Fix a number  $0 < s < 1$
- Divide  $s$  fraction of its pagerank value to its neighbours
- $1 - s$  fraction of its pagerank value to all nodes evenly

This is equivalent to the random walk. With probability  $s$ , we go to one of its neighbours and with probability  $1 - s$ , we go to a random page. The resulting graph is strongly connected and aperiodic, meaning it has a stationary distribution.

# Part III

## Mathematical Programming

# Chapter 9

## Linear Programming

### 9.1 Introduction

#### Definition: Mathematical Programming

Problems of the form

$$\begin{array}{ll}\min & f(x) \\ \text{s.t.} & g_1(x) \leq 0 \\ & \vdots \\ & g_m(x) \leq 0 \\ & x \in \mathbb{R}^n\end{array}$$

#### Definition: Affine Function

$f : \mathbb{R}^n \rightarrow \mathbb{R}$  where

$$f(x) = c_1x_1 + \cdots + c_nx_n + b = c^T x + b$$

#### Definition: Linear Programming

A mathematical programming problem where  $f, g_1, \dots, g_m$  are linear functions.

$$\begin{array}{ll}\min & c^T x \\ \text{s.t.} & Ax \leq b \\ & x \in \mathbb{R}^n\end{array}$$

where we can change  $A = \begin{pmatrix} A_1 & A_2 & \cdots & A_m \end{pmatrix}, b = \begin{pmatrix} b_1 & b_2 & \cdots & b_m \end{pmatrix}$  and the constraints are  $Ax \leq b$ .



**Definition: Standard Form**

$$\begin{aligned}
\min \quad & c^T x \\
\text{s.t.} \quad & Ax = b \\
& x \geq 0
\end{aligned}$$

Stock portfolio optimization:  $n$  companies, stock of company  $i$  costs  $c_i \in \mathbb{R}$ . Each company  $i$  has expected profit  $p_i \in \mathbb{R}$  and the budget is  $B \in \mathbb{R}$ .

$$\begin{aligned}
\max \quad & p^T x \\
\text{s.t.} \quad & c^T x \leq B \\
& x \geq 0
\end{aligned}$$

## 9.2 Fundamental Theorem of Linear Inequalities

**Theorem (Farkas (1894/1898)/Minkowski (1896))**

Let  $a_1, \dots, a_m, b \in \mathbb{R}^n$  and  $t = \text{rank}(a_1, \dots, a_m, b)$ . Then either

1.  $b$  is a convex combination of linearly independent vectors from  $a_1, \dots, a_m$ .
2. There exists a hyperplane  $H = \{x : c^T x = 0\}$  such that
  - $c^T b < 0$
  - $c^T a_i \geq 0$
  - $H$  contains  $t - 1$  linearly independent vectors from  $a_1, \dots, a_m$

**Definition: Separating Hyperplane**

The hyperplane  $H$  in the theorem is a separating hyperplane.

**Lemma (Farkas' Lemma)**

Let  $A \in \mathbb{R}^{m \times n}$  and  $b \in \mathbb{R}^m$ . The following are equivalent:

- There exists  $x \in \mathbb{R}^n$  such that  $x \geq 0$  and  $Ax = b$ .
- $y^T b \geq 0$  for each  $y \in \mathbb{R}^m$  such that  $y^T A \geq 0$ .

### Lemma (Farkas' Lemma - Variant 1)

Let  $A \in \mathbb{R}^{m \times n}$  and  $b \in \mathbb{R}^m$ . Then exactly one of the following statements hold:

- There exists  $x \in \mathbb{R}^n$  such that  $x \geq 0$  and  $Ax = b$ .
- There exists  $y \in \mathbb{R}^m$  such that  $y^T b \geq 0$  and  $y^T A \leq 0$ .

### Lemma (Farkas' Lemma - Variant 2)

Let  $A \in \mathbb{R}^{m \times n}$  and  $b \in \mathbb{R}^m$ . The following are equivalent:

- There exists  $x \in \mathbb{R}^n$  such that  $Ax \leq b$ .
- $y^T b \geq 0$  for each  $y \geq 0$  such that  $y^T A = 0$ .

**Proof.** Let  $M = \begin{bmatrix} I & A & -A \end{bmatrix}$ . Then  $Ax \leq b$  has a solution if and only if  $Mz = b$  has a nonnegative solution  $z \geq 0$ . Now apply the original version of the lemma.

## 9.3 Duality Theory

From Farkas' lemma, we have that  $Ax = b$  and  $x \geq 0$  if and only if  $y^T b \geq 0$  for each  $y \in \mathbb{R}^m$  such that  $y^T A \geq 0$ .

$$\begin{aligned} y^T A \leq c^T &\implies y^T Ax \leq c^T x \\ &\implies y^T b \leq c^T x \end{aligned}$$

Thus, if  $y^T A \leq c^T$ , then we have  $y^T b$  is a lower bound on the solution to the linear program.

Consider the primal LP  $\min\{c^T x : Ax = b, x \geq 0\}$  and the dual LP  $\max\{y^T b : y^T A \leq c^T\}$ .

### Theorem (Weak Duality)

Let  $x$  be a feasible solution of the primal LP (minimization) and  $y$  be a feasible solution of the dual LP (maximization). Then

$$y^T b \leq c^T x$$

Let  $\alpha^*, \beta^* \in \mathbb{R}$  be the optimal values for the primal and dual respectively.

- If the primal and dual are feasible, then

$$\beta^* \leq \alpha^*$$

- If the primal is **unbounded**, then the dual is **infeasible**.
- If the dual is **unbounded**, then the primal is **infeasible**.

**Theorem (Strong Duality)**

If the primal LP and dual LP are feasible, then

$$b^T y = \beta^* = \alpha^* = c^T x$$

**Proof.** Consider the LP

$$\begin{aligned} \min \quad & 0 \\ \text{s.t.} \quad & y^T A \leq c^T \\ & c^T x - y^T b \leq 0 \\ & Ax = b \\ & x \geq 0 \end{aligned}$$

The LP is encoded by

$$B \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} A & 0 \\ -A & 0 \\ c^T & -b^T \\ 0 & A^T \\ -I & 0 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \leq \begin{pmatrix} b \\ -b \\ 0 \\ c \\ 0 \end{pmatrix}$$

Farkas' lemma (variant 2) gives the system has a solution if and only if for each  $z = (u^T, v^T, \lambda, w^T, \alpha^T) \geq 0$  such that  $zB = 0$  then we have  $u^T b - v^T b + w^T c \geq 0$  (inner product with the right hand vector).

If  $\lambda > 0$ , and  $zB = 0$  if and only if  $u^T A - v^T A + \lambda c^T - \alpha^T = 0$  or  $u^T A - v^T A + \lambda c^T \geq 0$  since  $\alpha \geq 0$ . Also,  $w^T A^T = \lambda b^T$  or  $Aw = \lambda b$ . Solving, we have

$$\lambda[(u^T - v^T)b + c^T w] \geq 0$$

If  $\lambda = 0$ , let  $x, y$  be feasible solutions. Then  $x \geq 0, Ax = b, y^T A \leq c^T$ . Thus,

$$c^T w \geq y^T Aw = 0 \geq (v^T - u^T)Ax = (v^T - u^T)b$$

**Lemma (Affine Farkas' Lemma)**

Let the system  $Ax \leq b$  have at least one solution, and suppose that the inequality

$$c^T x \leq \delta$$

holds whenever  $x$  satisfies  $Ax \leq b$ . Then for some  $\delta' \leq \delta$ , the linear inequality

$$c^T x \leq \delta'$$

is a nonnegative linear combination of the inequalities of  $Ax \leq b$ .

### 9.3.1 Complementary Slackness

#### Theorem (Complementary Slackness)

If the optima in both the primal and dual is finite and  $x, y$  are feasible solutions, then the following are equivalent:

- $x, y$  are optimal solutions to the primal and dual.
- $c^T x = y^T b$ .
- If  $x_i > 0$ , then the corresponding inequality  $y^T A_i \leq c_i$  is an equality;  $y^T A_i = c_i$ .

## 9.4 Applications of LP Duality

### 9.4.1 Game Theory - Minimax Theorems

Let there be two players Alice and Bob where each player has a finite set of strategies  $S_A = \{1, \dots, m\}$  and  $S_B = \{1, \dots, n\}$ . There are payoff matrices  $A, B \in \mathbb{R}^{m \times n}$  for Alice and Bob, respectively, where if Alice plays  $i$  and Bob plays  $j$ , then

- Alice gets  $A_{ij}$
- Bob gets  $B_{ij}$

Assume players are rational, i.e. want to maximize their payoffs.

#### Definition: Nash Equilibrium

A strategy profile  $(i, j)$  is called a Nash equilibrium if the strategy played by each player is optimal, given the strategy of the other player. That is

- $A_{ij} \geq A_{kj}$  for all  $k \in S_A$ .
- $B_{ij} \geq B_{il}$  for all  $l \in S_B$

**Battle of the Sexes Game:**  $(2, 1), (1, 2)$  are Nash equilibria.

	Football	Opera
Football	(2, 1)	(0, 0)
Opera	(0, 0)	(1, 2)

Table 9.1: Battle of the sexes Payoff Matrix

**Prisoner's Dilemma:** The number in each cell is for example the number of years of jail.  $(-5, 5)$  is the Nash equilibrium.

	Silent	Snitch
Silent	$(-1, -1)$	$(-10, 0)$
Snitch	$(0, -10)$	$(-5, -5)$

Table 9.2: Prisoner's Dilemma

**Definition: Mixed Strategy**

A mixed strategy is a probability distribution over a set of pure strategies  $S$ . If Alice's strategies are  $S_A = \{1, \dots, n\}$ , her mixed strategies are

$$\Delta_A := \{x \in \mathbb{R}^n : x \geq 0, \|x\|_1 = 1\}$$

Models situation where players choose their strategy "at random".

**Definition: Expected Gain**

Payoffs for each player in mixed strategy.

$(x, y)$  is the profile of mixed strategies used by Alice and Bob. We have

$$v_A(x, y) = \sum_{(i,j) \in S_A \times S_B} A_{ij} x_i y_j = x^T A y$$

$$v_B(x, y) = \sum_{(i,j) \in S_A \times S_B} B_{ij} x_i y_j = x^T B y$$

**Definition: (Mixed) Nash Equilibrium**

A strategy profile  $x \in \Delta_A, y \in \Delta_B$  is called a mixed Nash equilibrium if the strategy played by each player is optimal, given the strategy of the other player. That is

- $x^T A y \geq z^T A y$  for all  $z \in \Delta_A$ .
- $x^T B y \geq x^T B w$  for all  $w \in \Delta_B$ .

	Jump left	Jump right
Kick left	$(-1, 1)$	$(1, -1)$
Kick right	$(1, -1)$	$(-1, 1)$

Table 9.3: Penalty Kick

There is no pure Nash equilibrium.

**Definition: Zero-Sum Game**

Payoff matrices satisfy  $A = -B$ .

Since it is a zero-sum game,  $A = -B$ , then Alice wants to maximize her payoff, while Bob wants to maximize his payoff, but  $B = -A$ , so it is equivalent to minimizing his payoff.

### Theorem (Minimax Theorem)

In a zero-sum game, for any payoff matrix  $A \in \mathbb{R}^{m \times n}$ :

$$\max_{x \in \Delta_A} \min_{y \in \Delta_B} x^T A y = \min_{y \in \Delta_B} \max_{x \in \Delta_A} x^T A y$$

**Proof.** For given  $x \in \Delta_A$ ,

$$\min_{y \in \Delta_B} x^T A y = \min_{j \in S_B} (x^T A)_j$$

The left hand side can be written as

$$\begin{aligned} \max \quad & s \\ \text{s.t.} \quad & s \leq (x^T A)_j, \forall j \in S_B \\ & \sum_{i \in S_A} x_i = 1 \\ & x \geq 0 \end{aligned}$$

For given  $y \in \Delta_B$ ,

$$\max_{x \in \Delta_A} x^T A y = \max_{i \in S_A} (A y)_i$$

The right hand side can be written as

$$\begin{aligned} \min \quad & t \\ \text{s.t.} \quad & t \geq (A y)_i, \forall i \in S_A \\ & \sum_{j \in S_B} y_j = 1 \\ & y \geq 0 \end{aligned}$$

There programs are duals of each other.

**Proof.** (Dual) Use the second LP,

$$\begin{aligned} \min \quad & t_+ - t_- \\ \text{s.t.} \quad & t_+ - t_- w_i - (A y)_i = 0, \forall i \in S_A \\ & \sum_{j \in S_B} y_j = 1 \\ & y \geq 0 \end{aligned}$$

which is

$$\begin{aligned} \min \quad & (e_1 - e_2)^T z \\ \text{s.t.} \quad & C = \begin{bmatrix} 0 & 0 & I & 0 \\ 1 & -1 & -A & -I \end{bmatrix} \begin{bmatrix} t_+ \\ t_- \\ y \\ w \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \\ & z \geq 0 \end{aligned}$$

finding the dual results in the first LP

### 9.4.2 Learning Theory - Boosting

Consider the classification problem over  $\mathcal{X} = \{x_1, \dots, x_m\}$  where the set of hypothesis  $\mathcal{H} := \{h : \mathcal{X} \rightarrow \{0, 1\}\}$  and each  $x \in \mathcal{X}$  has a correct value  $c(x) \in \{0, 1\}$ . Data is sampled from an unknown distribution  $q \in \Delta_{\mathcal{X}}$ .

#### Weak Learning Assumption

For any distribution  $q \in \Delta_{\mathcal{X}}$ , there is a hypothesis  $h \in \mathcal{H}$  which is wrong less than half the time.

$$\exists \gamma > 0, \forall q \in \Delta_{\mathcal{X}}, \exists h \in \mathcal{H}, P_{x \sim q}[h(x) \neq c(x)] \leq \frac{1 - \gamma}{2}$$

Weak learning assumption implies something much stronger: it is possible to combine classifiers in  $\mathcal{H}$  to construct a classifier that is always right (strong learning)

#### Theorem

Let  $\mathcal{H}$  be a set of hypotheses satisfying weak learning assumption. Then there is a distribution  $p \in \Delta_{\mathcal{H}}$  such that the weighted majority classifier

$$c_p(x) := \begin{cases} 1 & \text{if } \sum_{h \in \mathcal{H}} p_h h(x) \geq \frac{1}{2} \\ 0 & \text{otherwise} \end{cases}$$

is always correct. That is,  $c_p(x) = c(x)$  for all  $x \in \mathcal{X}$ .

**Proof.** Let  $M \in \{-1, 1\}^{m \times n}$  where  $m = |\mathcal{X}|$  and  $n = |\mathcal{H}|$ .

$$M_{ij} = \begin{cases} +1 & \text{if classifier } h_j \text{ wrong on } x_i \\ -1 & \text{otherwise} \end{cases}$$

Weak learning:

$$\sum_{1 \leq i \leq m} q_i \cdot \delta_{h_j(x_i) \neq c(x_i)} \leq \frac{1 - \gamma}{2}$$

Note that  $M_{ij} = 2 \cdot \delta_{h_j(x_i) \neq c(x_i)} - 1$ , where  $\delta_{h_j(x_i) \neq c(x_i)} = 1$  if they are different and 0 if they are the same, and

$$q^T M e_j \leq -\gamma \implies q^T M p \leq -\gamma$$

for any  $p \in \Delta_{\mathcal{H}}$ . By minimax, we have

$$\max_{q \in \Delta_{\mathcal{X}}} \min_{p \in \Delta_{\mathcal{H}}} q^T M p = \min_{p \in \Delta_{\mathcal{H}}} \max_{q \in \Delta_{\mathcal{X}}} q^T M p \leq -\gamma$$

In particular, the right hand side implies weighted classifier given by optimal solution  $p^*$  is always correct.

**Proof.** Pick an image  $x_i$  is equivalent to choosing  $q = e_i$ . We know that  $e_i^T M p^* \leq -\gamma$ .

$$\sum_{j=1}^n M_{ij} p_j^* \leq -\gamma \iff \sum_{j=1}^n p_j^* \delta_{h_j(x_i) \neq c(x_i)} \leq \frac{1 - \gamma}{2}$$

### 9.4.3 Combinatorics - Bipartite Matching

Given a bipartite graph  $G = (L \cup R, E)$ , does it have a perfect matching? We previously saw that we can randomly isolate a perfect matching, but we want to remove the randomness. This would lead to a fast parallel algorithm for matchings.

#### Definition: Circulation

Given an even cycle  $C = (e_1, e_2, \dots, e_{2k})$ , the circulation of  $C$  is given by

$$\text{circ}(C) = |w(e_1) - w(e_2) + \dots + w(e_{2k-1}) - w(e_{2k})|$$

#### Lemma

If we assign weights  $w(e_i)$  such that  $\text{circ}(C) \neq 0$  for each cycle  $C$  of the bipartite graph  $G$ , then we get the minimum weight perfect matching is unique.

Suppose we have a weight assignment  $w$ . Let  $G_w$  be the subgraph of  $G$  given by the union of all min  $w$ -weight perfect matchings in  $G$ .

#### Claim

Circulation of each even cycle in  $G_w$  is zero.

**Proof.** Use LP duality. The primal problem is

$$\begin{aligned} \min \quad & \sum_{e \in E} w_e x_e \\ \text{s.t.} \quad & \sum_{e \in \delta(u)} x_e = 1, \quad \forall u \in L \cup R \\ & x \geq 0 \end{aligned}$$

The dual is

$$\begin{aligned} \max \quad & \sum_{u \in L \cup R} y_u \\ \text{s.t.} \quad & y_u + y_v \leq w_e, \quad \forall e = (u, v) \in E \end{aligned}$$

By complementary slackness,  $x_e \neq 0$  in the primal means  $y_u + y_v = w_e$  is the dual to be optimal.



# Chapter 10

## Semidefinite Programming

### 10.1 Positive Semidefinite Matrices

**Definition: Symmetric Matrix**

A matrix  $S \in \mathbb{R}^{n \times n}$  is symmetric if  $S_{ij} = S_{ji}$  for all  $i, j \in [n]$ .

**Definition: Spectral Theorem**

Any symmetric matrix in  $\mathbb{R}^{n \times n}$  has  $n$  real eigenvalues and an orthonormal basis in  $\mathbb{R}^n$  for the eigenvectors, i.e. we can write

$$S = \sum_{i=1}^n \lambda_i v_i v_i^T$$

where  $\lambda_i \in \mathbb{R}$  and  $v_i \in \mathbb{R}^n$  such that  $\langle v_i, v_j \rangle = \delta_{ij}$ .

**Definition: Positive Semidefinite (PSD)**

A symmetric matrix  $S \in \mathbb{R}^{n \times n}$  having only nonnegative eigenvalues. We write  $S \succeq 0$ .

### Lemma (Equivalent Characterizations of PSD Matrices)

- All eigenvalues of  $S$  are nonnegative.
- $S = Y^T Y$  for some  $Y \in \mathbb{R}^{d \times n}$  where  $d \leq n$  (smallest  $d$  is equal to rank).
- $x^T S x \geq 0$  for all  $x \in \mathbb{R}^n$ .
- $S = LDL^T$ , where  $D$  is a nonnegative diagonal matrix and  $L$  is the unit lower-triangular matrix (only 1's along diagonal, other values in the lower-triangular part).
- $S$  is in the convex hull of the set

$$\{uu^T : u \in \mathbb{R}^n\}$$

- $S = U^T D U$  where  $D$  is a nonnegative diagonal matrix and  $U \in \mathbb{R}^{n \times n}$  is an orthonormal matrix ( $U^T U = I$ ).
- Any principal minor of  $A$  has nonnegative determinant (choose  $A \subseteq [n]$ , and remove all rows/columns not in  $A$ ).

## 10.2 Semidefinite Programming Formulation

Let  $\mathcal{S}^m := \mathcal{S}^m(\mathbb{R})$  be the space of all  $m \times m$  real symmetric matrices.

### Definition: Semidefinite Programming (SDP)

Deals with problems of the form

$$\begin{aligned} \min \quad & c^T x \\ \text{s.t.} \quad & x_1 \cdot A_1 + \cdots + x_n \cdot A_n \succeq B \\ & x \in \mathbb{R}^n \\ & A_i, B \in \mathcal{S}^m(\mathbb{R}) \end{aligned}$$

$C \succeq D$  denotes that  $C - D \succeq 0$ , i.e.  $C - D$  is PSD.

### 10.2.1 LP vs. SDP

Let the LP  $\min\{a^T x : Cx \geq b : x \in \mathbb{R}^n\}$  and the SDP  $\min\{c^T x : x_1 \cdot A_1 + \cdots + x_n \cdot A_n \succeq B, x \in \mathbb{R}^n\}$ . If  $(Ax)_i \geq b_i$ , we can set  $A_j$ 's to be diagonal matrices where each entry in

diagonal is  $A_{ij}$  and  $B = \text{diag}(b_1, \dots, b_m)$ . Then

$$\sum_j x_j A_j - B = \text{diag}(\sum A_{1j} x_j - b_1, \dots, \sum A_{mj} x_j - b_m)$$

## 10.3 Convex Algebraic Geometry

### Definition: Linear Matrix Inequalities (LMI)

A linear matrix inequality is an inequality of the form

$$A_0 + \sum_{i=1}^n A_i x_i \succeq 0$$

where  $A_0, \dots, A_n$  are symmetric matrices.

### Definition: Spectrahedron

A set defined by finite many linear matrix inequality.

$$S = \left\{ x \in \mathbb{R}^n : \sum_{i=1}^n A_i x_i \succeq B, A_i, B \in \mathcal{S}^m \right\}$$

It turns out, it is a set defined by only one LMI. If we had  $\sum_i A_i x_i \succeq B$  and  $\sum_i C_i x_i \succeq D$ , then  $E_i = \begin{bmatrix} A_i & 0 \\ 0 & C_i \end{bmatrix}$  and  $F = \begin{bmatrix} B & 0 \\ 0 & D \end{bmatrix}$ , and this all turns into

$$\sum_i E_i x_i \succeq F \implies \begin{bmatrix} \sum_i A_i x_i - B & 0 \\ 0 & \sum_i C_i x_i - D \end{bmatrix}$$

**Examples of spectrahedra:**

- Circle:  $\mathcal{C} = \{(x, y) \in \mathbb{R}^2 : x^2 + y^2 \leq 1\}$ . Then

$$\begin{bmatrix} 1-x & y \\ y & 1+x \end{bmatrix} \succeq 0$$

Principal minors determinants:  $1-x \geq 0, 1+x \geq 0, 1-x^2-y^2 \geq 0$ .

- Hyperbola:  $\mathcal{H} = \{(x, y) \in \mathbb{R}^2 : xy \geq 1, x \geq 0, y \geq 0\}$ . Then

$$\begin{bmatrix} x & 1 \\ 1 & y \end{bmatrix}$$

Principal minors determinants:  $xy - 1 \geq 0, x \geq 0, y \geq 0$ .

- Elliptic Curve:  $\mathcal{E} = \{(x, y) \in \mathbb{R}^2 : -2y^2 - x^3 - 3x^2 + x + 3 = 0\}$ . Then

$$\begin{bmatrix} x+1 & 0 & y \\ 0 & 2 & -x-1 \\ y & -x-1 & 2 \end{bmatrix}$$

Principal minors determinants:  $x+1 \geq 0, 4-(x+y)^2 \geq 0, 2x+2-y^2 \geq 0, -2y^2-x^3-3x^2+x+3 \geq 0$ .

### Definition: Projected Spectrahedron

A set  $S \in \mathbb{R}^n$  that has the form

$$S = \left\{ x \in \mathbb{R}^n : \exists y \in \mathbb{R}^t \text{ s.t. } \sum_{i=1}^n A_i x_i + \sum_{j=1}^t B_j y_j \succeq C, A_i, B_j, C \in \mathcal{S}^m \right\}$$

**Examples of projected spectrahedra:**

- Projected hyperbola:  $\mathbb{R}_{>0} = \left\{ x \in \mathbb{R} : \exists y \in \mathbb{R}, \begin{bmatrix} x & 1 \\ 1 & y \end{bmatrix} \succeq 0 \right\}$ .
- Projected quadratic cone intersected with halfspace:

$$S = \left\{ (x, y) \in \mathbb{R}^2 : \exists z \in \mathbb{R}, \begin{bmatrix} z+y & 2z-x \\ 2z-x & z-y \end{bmatrix} \succeq 0 \right\}$$

### 10.3.1 Testing Points in Spectrahedron

To be able to optimize, we must be able to test whether a point  $x \in \mathbb{R}^n$  is inside our spectrahedron

$$S = \left\{ x \in \mathbb{R}^n : \sum_{i=1}^n A_i x_i \succeq B, A_i, B \in \mathcal{S}^m \right\}$$

By definition of  $x \in S$ , it is equivalent to saying  $Z = \sum_{i=1}^n A_i x_i - B \succeq 0$ . We can use

**Symmetric Gaussian Elimination.**

Let  $Z = \begin{bmatrix} z_{11} & z_{12} & \cdots & z_{1n} \\ z_{12} & & & \\ \vdots & & & \\ z_{1n} & & & \end{bmatrix}$ ,  $L_1 = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ -z_{12}/z_{11} & 1 & \cdots & 0 \\ \vdots & 0 & \cdots & 0 \\ -z_{1n}/z_{11} & 0 & \cdots & 1 \end{bmatrix}$ . Then  $L_1 Z L_1^T = \text{diag}(z_{11}, Z_1)$  is

symmetric. If  $z_{11} = 0$ , then the first row and column must all be 0, otherwise  $Z$  is not PSD.

This algorithm has strongly polynomial runtime.

## 10.4 Control Theory

### Definition: Linear Difference Equation

$$x(t+1) = Ax(t), \quad x(0) = x_0$$

When  $A$  and  $x_0$  are nonnegative, this is a Markov chain.

When  $t \rightarrow \infty$ , under what conditions will  $x(t)$  remain bounded? Or go to zero?

### Definition: Stable

When the system  $x(t)$  converges to zero.

### Proposition

The system  $x(t)$  is stable if and only if  $|\lambda_i(A)| < 1$ .

### Definition: Lyapunov Functions

Generalize energy in systems. Functions on  $x(t)$  decrease monotonically on trajectories of the system.

For discrete-time system,

$$V(x(t)) = x(t)^T P x(t)$$

To make it monotonically decreasing, we need

$$\begin{aligned} V(x(t+1)) \leq V(x(t)) &\iff x(t+1)^T P x(t+1) - x(t)^T P x(t) \leq 0 \\ &\iff x(t)^T A^T P A x(t) - x(t)^T P x(t) \leq 0 \\ &\iff A^T P A - P \preceq 0 \end{aligned}$$

### Theorem

Given a matrix  $A \in \mathbb{R}^{m \times m}$ , the following conditions are equivalent:

1. All eigenvalues of  $A$  are inside the unit circle, i.e.  $|\lambda_i(A)| < 1$ .
2. There is  $P \in \mathcal{S}^m$  such that  $P \succ 0$  and  $A^T P A - P \prec 0$ .

**Proof.** ( $\implies$ ) Let  $P = \sum_{i=1}^{\infty} (A A^T)^i \succ 0$ .

$$A^T P A - P = \sum_{i=1}^{\infty} (A^T A)^i - \sum_{i=0}^{\infty} (A A^T)^i = -I + \sum_{i \geq 1} (A^T A)^i - \sum_{i \geq 1} (A A^T)^i \prec 0$$

( $\impliedby$ ) We know  $A^T P A - P \prec 0$ . Then  $x^T (A^T P A - P) x < 0$ . If  $x = v_i$  be the eigenvector corresponding to eigenvalue  $\lambda_i$ . Then,

$$(Ax)^T P (Ax) - x^T P x < 0 \implies \lambda_i v_i^T P \lambda_i v_i - v_i^T P v_i < 0 \implies (|\lambda_i|^2 - 1) v_i^T P v_i < 0$$

Since  $P$  is positive definite, then  $v_i^T P v_i > 0$ , then  $|\lambda_i| < 1$  for the inequality to hold.

**Definition: Linear Difference Equation with Control Input**

Let  $A \in \mathbb{R}^{m \times m}, B \in \mathbb{R}^{m \times k}$ ,

$$x(t+1) = Ax(t) + Bu(t), \quad x(0) = x_0$$

If we properly choose control input  $u(t)$ , we can make our system  $x(t)$  behave in a way that we want. We want to set the control input to be  $u(t) = Kx(t)$  for some fixed  $K$ . This is equivalent as replacing  $A$  with  $A + BK$ . This is harder to solve via simple eigenvalue description, but still solved in the same way via Lyapunov functions. We want  $P \succ 0$  such that

$$(A + BK)^T P (A + BK) - P \prec 0$$

We can make this into an SDP with some matrix manipulations.

## 10.5 Duality Theory

**Definition: Frobenius Inner Product**

Let  $A, B \in \mathcal{S}^m$ , then

$$\langle A, B \rangle := \text{tr}[AB] = \sum_{i,j} A_{ij} B_{ij}$$

**Definition: Frobenius Norm**

$$\|A\|_F = \sqrt{\langle A, A \rangle} = \sqrt{\sum_{i,j} A_{ij}^2}$$

**Definition: Polar Dual**

Given a spectrahedron  $S \subseteq \mathcal{S}^m$ ,

$$S^\circ = \{Y \in \mathcal{S}^m : \langle Y, X \rangle \leq 1, \forall X \in S\}$$

**Definition: Standard Semidefinite Program**

$$\begin{aligned} \min \quad & \langle C, X \rangle \\ \text{s.t.} \quad & \langle A_i, X \rangle = b_i \\ & X \succeq 0 \end{aligned}$$

The variables are a positive semidefinite matrix  $X$ , each constraint is given by the Frobenius inner product  $\langle A_i, X \rangle = b_i$  and we can obtain the LP standard form by making  $X$  and  $A_i$ 's

to be diagonal.

**Standard Primal as LMI:** Since  $\langle A_i, X \rangle = \sum_{j \leq k} A_{ijk} x_{jk}$ , then  $L = \text{diag}(\sum A_{1jk} x_{jk}, \dots, \sum A_{tjk} x_{jk})$ ,

$$\begin{bmatrix} L - \text{diag}(b) & 0 & 0 \\ 0 & -L + \text{diag}(b) & 0 \\ 0 & 0 & X \end{bmatrix}$$

$$\begin{aligned} \min \quad & \langle C, X \rangle \\ \text{s.t.} \quad & \sum_{i,j} \Gamma_{ij} x_{ij} + \Gamma_0 \succeq 0 \\ & X \in \mathbb{R}^{\binom{m+1}{2}} \end{aligned}$$

$$\Gamma_0 = \begin{bmatrix} -\text{diag}(b) & 0 & 0 \\ 0 & \text{diag}(b) & 0 \\ 0 & 0 & 0 \end{bmatrix} \text{ and } \Gamma_{jk} = \text{diag}(\text{diag}(A_{ijk}), -\text{diag}(A_{ijk}), E_{jk} + E_{kj}).$$

### Proposition

If  $X \succeq 0$  and  $A \succeq B$ , then  $\langle A, X \rangle \succeq \langle B, X \rangle$ .

Consider the SDP and multiply  $i$ th equality by variable  $y_i$ :

$$\sum_{i=1}^t y_i \langle A_i, X \rangle = \sum_{i=1}^t y_i b_i \implies \left\langle \sum_{i=1}^t y_i A_i, X \right\rangle = y^T b$$

Thus, if  $\sum_{i=1}^t y_i A_i \preceq C$ , then

$$y^T b = \left\langle \sum_{i=1}^t y_i A_i, X \right\rangle \leq \langle C, X \rangle$$

Therefore,  $y^T b$  is a lower bound on the SDP.

### Definition: Dual Semidefinite Program

$$\begin{aligned} \max \quad & y^T b \\ \text{s.t.} \quad & \sum_{i=1}^t y_i A_i \preceq C \end{aligned}$$

### Theorem (Weak Duality)

Let  $X$  be a feasible solution of the primal SDP and  $y$  be a feasible solution of the dual SDP. Then

$$y^T b \leq \langle C, X \rangle$$

**Theorem (Complementary Slackness)**

Let  $X$  be a feasible solution of the primal SDP and  $y$  be feasible solution of the dual SDP. If  $(X, y)$  satisfy the complementary slackness condition

$$\left(C - \sum_{i=1}^t y_i A_i\right) X = 0$$

Then  $(X, y)$  are primal and dual optimum solutions of the SDP problem.

Strong duality for SDPs are more complex than LPs; both the primal and dual may be feasible, but strong duality may not hold.

**Definition: Strictly Feasible**

The primal SDP is strictly feasible if there is a feasible solution  $X \succ 0$ .

The dual SDP is strictly feasible if there is a feasible solution  $y$  such that  $\sum_{i=1}^t y_i A_i \prec C$ .

**Theorem (Strong Duality under Slater Conditions)**

If the primal SDP and dual SDP are both strictly feasible, then the optimal value of the primal equals the optimal value of the dual.



## Part IV

# Approximation Algorithms

# Chapter 11

## Linear Programming Relaxations and Rounding

Many optimization problems are NP-hard. We want to find approximate solutions in polynomial time and even for problems in P.

### Definition: Integer Linear Program (ILP)

$$\begin{array}{ll}\min & c^T x \\ \text{s.t.} & Ax \leq b \\ & x \in \mathbb{N}^n\end{array}$$

ILPs capture many combinatorial optimization problems, but also capture NP-hard problems.

### 11.1 Independent Set

#### Maximum Independent Set Problem

Let  $G = (V, E)$  be a graph. Find an maximal independent set  $S \subseteq V$  such that  $u, v \in S \implies \{u, v\} \notin E$ .

### Maximum Independent Set ILP

$$\begin{aligned} \max \quad & \sum_{v \in V} x_v \\ \text{s.t.} \quad & x_u + x_v \leq 1, \forall \{u, v\} \in E \\ & x_v \in \{0, 1\}, \forall v \in V \end{aligned}$$

To get efficient (exact or approximate) algorithms for problems, the following strategy is useful:

1. Formulate combinatorial optimization problem as ILP.
2. Derive LP from ILP by removing integral constraints (LP relaxation).
3. We are minimizing over the same objective function, by over a potentially larger set of solutions.

$$\text{opt(LP)} \leq \text{opt(ILP)}$$

4. Solve the LP using an efficient algorithm. If the solution is integral, then we have our optimal solution, otherwise, if we have fractional values, we have to round our answer by some procedure so that

$$\text{opt(LP)} \leq \text{rounded solution} \leq c \cdot \text{opt(ILP)}$$

where  $c$  is our *approximation* factor

Let  $P = \{x \in \mathbb{R}_{\geq 0}^n : Ax = b\}$  be the polytope defined by the LP  $\min\{c^T x : Ax = b, x \geq 0\}$ .

#### Definition: Vertex Solutions

A solution  $x \in P$  is a vertex solution if  $\nexists y \neq 0$  such that  $x + y \in P$  and  $x - y \in P$ .

#### Definition: Extreme Point Solutions

$x \in P$  is an extreme point solution if  $\exists u \in \mathbb{R}^n$  such that  $x$  is the unique optimum solution to the LP with constraint  $P$  and objective  $u^T x$ .

#### Definition: Basic Solutions

Let  $\text{supp}(x) := \{i \in [n] : x_i > 0\}$  be the set of nonzero coordinates of  $x$ . Then  $x \in P$  is a basic solution if and only if the columns of  $A$  indexed by  $\text{supp}(x)$  are linearly independent.

## 11.2 Vertex Cover

### Minimum Vertex Cover Problem

**Input:** graph  $G = (V, E)$  (can have cost  $c_v$  for all  $v \in V$ ).

**Output:** minimum set  $S \subseteq V$  such that for each  $\{u, v\} \in E$ , we have

$$|S \cap \{u, v\}| \geq 1$$

**Weighted Version:** each vertex  $v \in V$  has a cost  $c_v \in \mathbb{R}_{\geq 0}$ .

### Vertex Cover ILP

$$\begin{aligned} \min \quad & \sum_{u \in V} c_u x_u \\ \text{s.t.} \quad & x_u + x_v \geq 1, \quad \forall \{u, v\} \in E \\ & x_u \in \{0, 1\}, \quad \forall u \in V \end{aligned}$$

### 11.2.1 Unweighted Simple 2-Approximation

- 
- 1: List edges  $E$  in any order
  - 2:  $S = \emptyset$
  - 3: **for**  $\{u, v\} \in E$  **do**
  - 4:     **if**  $S \cap \{u, v\} = \emptyset$  **then**
  - 5:          $S = S \cup \{u, v\}$
  - 6: **return**  $S$
- 

**Correctness:** By construction  $S$  is a vertex cover. If added elements to  $S$   $k$  times, then  $|S| = 2k$  and  $G$  has a matching of size  $k$ , which means that optimum vertex cover is at least  $k$ . So we get a 2-approximation.

### 11.2.2 LP Relaxation

Drop the integrality constraints

$$\begin{aligned} \min \quad & c^T x \\ \text{s.t.} \quad & x_u + x_v \geq 1, \quad \forall \{u, v\} \in E \\ & 0 \leq x_u \leq 1, \quad \forall u \in V \end{aligned}$$

Solve this LP and get the optimal solution  $z$  and round  $z_v$  to the nearest integer to get the solution  $y$ . Since each edge is covered, at least one of  $z_u, z_v$  is  $\geq 1/2$  by the feasibility of LP.

Cost of  $y$  is

$$\sum_{u \in V} c_u y_u \leq \sum_{u \in V} c_u (2z_u) \leq 2OPT(ILP)$$

## 11.3 Set Cover

### Minimum Set Cover Problem

**Input:** a finite set  $U$  and a collection  $S_1, \dots, S_n$  of subsets of  $U$ .

**Output:** fewest collection of sets  $I \subseteq [n]$  such that

$$\bigcup_{j \in I} S_j = U$$

**Weighted version:** each set  $S_i$  has a weight  $w_i \in \mathbb{R}_{\geq 0}$ .

### Set Cover ILP

$$\begin{aligned} \min \quad & \sum_{i \in [n]} w_i x_i \\ \text{s.t.} \quad & \sum_{i: v \in S_i} x_i \geq 1, \quad \forall v \in U \\ & x_i \in \{0, 1\}, \quad \forall i \in [n] \end{aligned}$$

### 11.3.1 LP Relaxation

$$\begin{aligned} \min \quad & \sum_{i \in [n]} w_i x_i \\ \text{s.t.} \quad & \sum_{i: v \in S_i} x_i \geq 1, \quad \forall v \in U \\ & 0 \leq x_i \leq 1, \quad \forall i \in [n] \end{aligned}$$

Suppose we get a feasible solution  $z \in [0, 1]^n$ , can we just round each  $z_i$  to the nearest integer?

No, say  $v \in U$  is in 20 sets and  $z_i = \frac{1}{20}$  for each of the sets  $v \in S_i$ , then rounding would not select any  $S_i$  where  $v \in S_i$ .

### 11.3.2 Random Pick

Think of  $z_i$  as the probability we would pick  $S_i$ .  $z$  describes an optimal probability distribution over ways to choose  $S_i$ .

---

**Algorithm 4** Random Pick

---

1: **Input:**  $z = (z_1, \dots, z_n) \in [0, 1]^n$  such that  $z$  is optimal to LP.  
2: **Output:** set cover for  $U$   
3:  $I = \emptyset$   
4: **for**  $i = 1$  to  $n$  **do**  
5:      $I = I \cup \{i\}$  with probability  $z_i$   
6: **return**  $I$

---

Consider the random pick process from point of view  $v \in U$ . Let  $v \in S_1, \dots, S_k$ . We select  $S_i$  with probability  $z_i$  such that  $\sum_{i=1}^k z_i \geq 1$ .

For example, let  $v \in S_1, S_2$  and probability of picking  $S_i$  is  $P[S_i] = \frac{1}{2}$ .

$$P[v \text{ not covered}] = P[\neg S_1] \cdot P[\neg S_2] = (1 - z_1)(1 - z_2) = \frac{1}{4}$$

Then the probability of being covered is

$$P[v \text{ covered}] = 1 - \frac{1}{4} = \frac{3}{4}$$

**Lemma (Probability of Covering an Element)**

In a sequence of  $k$  independent experiments, in which the  $i$ th experiment has success probability  $p_i$  and

$$\sum_{i=1}^k p_i \geq 1$$

then there is a probability  $\geq 1 - 1/e$  that at least one experiment is successful.

**Proof.** Probability that we fail is  $(1 - p_1) \cdots (1 - p_k)$ . Note that  $1 - x \leq e^{-x}$  so  $(1 - p_1) \cdots (1 - p_k) \leq e^{-\sum p_i} \leq e^{-1}$ .

## 11.4 Randomized Rounding

---

1: **Input:**  $z = (z_1, \dots, z_n) \in [0, 1]^n$  such that  $z$  is a solution to our LP  
2: **Output:** set cover for  $U$   
3:  $I = \emptyset$   
4: **while** there is  $v \in U$  uncovered **do**  
5:     **for**  $i = 1$  to  $n$  **do**  
6:          $I = I \cup \{i\}$  with probability  $z_i$   
7: **return**  $I$

---

**Lemma (Probability Decay)**

Let  $t \in \mathbb{N}$ . The probability that the for loop will be executed more than  $\ln(|U|) + t$  times is at most  $e^{-t}$ .

**Proof.** The probability that  $j$  is not covered is at most  $e^{-(t+\log m)}$  by previous lemma. By union bound on all  $m$  elements is equal to  $\frac{1}{m}e^{-t}$  which is at most  $e^{-t}$ .

**11.4.1 Cost of Rounding**

We now know will cover with high probability, so we just need to bound the cost of the solution. At each iteration of the for loop, our expected cover weight is

$$\sum_{i=1}^n w_i z_i$$

After  $t$  iterations of the for loop, the expected weight is

$$\omega = t \sum_{i=1}^n w_i z_i$$

By Markov,

$$P[X \geq 2\mathbb{E}[X]] \leq \frac{1}{2}$$

**Lemma (Cost of Rounding)**

Given  $z$  optimal for the LP, our randomized rounding outputs, with probability  $\geq 0.45$ , a feasible solution to set cover with  $\leq 2(\ln |U| + 3) \cdot \text{OPT(ILP)}$  cost.

**Proof.** Let  $t = \ln(|U|) + 3$ . There is a probability at most  $e^{-3} < 0.05$  that the while loop runs more than  $t$  steps. After  $t$  steps, the expected weight is

$$\omega = t \sum w_i z_i \leq t \text{OPT(ILP)}$$

Markov's inequality shows that probability that our solution has weight  $\geq 2\omega$  is  $\leq \frac{1}{2}$ . By union bound, with probability  $\leq 0.55$ , either run for more than  $t$  times, or our solution has weight  $\geq 2\omega$ . Thus, with probability  $\geq 0.45$ , we stop at  $t$  iterations and construct a solution to set cover with cost  $\leq 2t \text{OPT(ILP)}$ .

# Chapter 12

## Semidefinite Programming Relaxations and Rounding

### Definition: Quadratic Program (QP)

$$\begin{array}{ll}\min & g(x) \\ \text{s.t.} & q_i(x) \geq 0\end{array}$$

where  $g(x)$  and each  $q_i(x)$  are quadratic functions on  $x$ .

QPs can formulate optimization problems but can capture NP-hard problems. In an ILP, if  $x \in \{0, 1\}$ , then the quadratic constraint is  $x(x - 1) = 0$ .

We can relax QPs to SDPs to get better approximation algorithms.

### 12.1 Maximum Cut

#### Maximum Cut Problem

Given a graph  $G = (V, E)$ , find a cut  $S \subseteq V$  of maximum size.

The ILP is

$$\begin{array}{ll}\max & \sum_{e \in E} w_e z_e \\ \text{s.t.} & x_u + x_v \geq z_e, \forall e = \{u, v\} \in E \\ & 2 - x_u - x_v \geq z_e, \forall e = \{u, v\} \in E \\ & x_v \in \{0, 1\}, \forall v \in V\end{array}$$

To get efficient exact or approximate algorithms for problems, the following strategy is useful:



1. Formulate the optimization problem as a quadratic program.
2. Derive the SDP from the QP by going to higher dimensions and imposing the PSD constraint; this is called an SDP relaxation.
3. We are still maximizing the same objective function but over a potentially larger set of solutions, thus,

$$OPT(SDP) \geq OPT(QP)$$

4. Solve the SDP approximately using an efficient algorithm.
  - If the solution to SDP is integral and one-dimensional, then it is a solution to QP and we are done.
  - If the solution has higher dimension, then we have to devise a rounding procedure that transforms it to an integral and one-dimensional solution. The rounded SDP solution  $\geq c \cdot OPT(QP)$ .

Suppose  $\sum_{e \in E} = 1$  in the problem.

**Proposition**

$OPT(ILP) = 1$  if and only if  $G$  is bipartite.

**Proposition**

$OPT(ILP) \geq \frac{1}{2}$ .

**Proof.**  $z_e = 1$  with probability  $\frac{1}{2}$  and  $z_e = 0$  with probability  $\frac{1}{2}$ .

$$\begin{aligned} \mathbb{E}_{x_u \sim \{0,1\}} [E(S, \bar{S})] &= \sum_{e \in E} w_e \mathbb{E}[z_e] \\ &= \sum_{e \in E} w_e \cdot \frac{1}{2} \\ &= \frac{1}{2} \end{aligned}$$

**Proposition**

If  $G$  is a complete graph, then  $OPT = \frac{1}{2} + \frac{1}{2(n-1)}$ .

The LP relaxation:

$$\begin{aligned}
\max \quad & \sum_{e \in E} w_e z_e \\
\text{s.t.} \quad & x_u + x_v \geq z_e, \quad \forall e = \{u, v\} \in E \\
& 2 - x_u - x_v \geq z_e, \quad \forall e = \{u, v\} \in E \\
& 0 \leq x_v \leq 1, \quad \forall v \in V \\
& 0 \leq z_e \leq 1, \quad \forall e \in E
\end{aligned}$$

Setting  $x_v = \frac{1}{2}$ ,  $z_e = 1$ , we always get  $OPT(LP) = 1$ , thus this is not helpful.

QP: Consider  $1 - (x_u x_v + (1 - x_u)(1 - x_v))$ , this is 1 if  $x_u \neq x_v$  and 0 if  $x_u = x_v$  which is selecting an edge when  $x_u$  is in the other side of the cut to  $x_v$ .

$$\begin{aligned}
\max \quad & \sum_{e=\{u,v\} \in E} w_e (-2x_u x_v + x_u + x_v) \\
\text{s.t.} \quad & x_v(x_v - 1) = 0, \quad \forall v \in V
\end{aligned}$$

If we instead choose that  $x_v = \{-1, 1\}$  instead of  $\{0, 1\}$ , we can get a better looking program:

$$\begin{aligned}
\max \quad & \sum_{e=\{u,v\} \in E} \frac{w_e}{2} (1 - x_u x_v) \\
\text{s.t.} \quad & x_v^2 = 1, \quad \forall v \in V
\end{aligned}$$

We can transform the QP into higher dimension relaxation with variable  $y_v \in \mathbb{R}^d$  and get the SDP relaxation (Delorme, Poljak 1993):

$$\begin{aligned}
\max \quad & \sum_{e=\{u,v\} \in E} \frac{w_e}{2} (1 - \langle y_u, y_v \rangle) \\
\text{s.t.} \quad & \|y_v\|^2 = 1, \quad \forall v \in V
\end{aligned}$$

The SDP is

$$\begin{aligned}
\max \quad & \sum_{\{u,v\} \in E} w_{uv} \left( \frac{1 - X_{uv}}{2} \right) \\
\text{s.t.} \quad & X_{uv} = 1 \\
& X \preceq 0
\end{aligned}$$

where  $X = Y^T Y$  and  $Y$  is a matrix where each column is  $y_i$ .

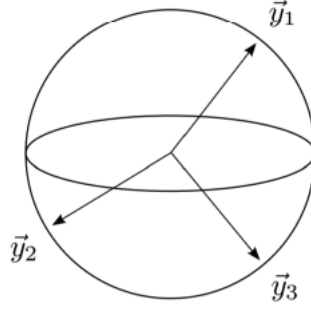
Note that the constraints form a unit sphere in  $\mathbb{R}^d$ .

### 12.1.1 SDP Explanation

Let  $\gamma_{uv} = y_u^T y_v = \cos(y_u, y_v)$ . For any edge, we want  $\gamma_{uv} \approx -1$  as this maximizes the weight. Geometrically, we want vertices from our max-cut  $S$  to be as far away from  $\bar{S}$  as possible. If all  $y_v$ 's are in one-dimensional space, then we get original quadratic program

$$OPT(SDP) \geq \text{Weight of max-cut}$$

Figure 12.1: Vectors  $\vec{y}_v$  embedded onto a unit sphere in  $\mathbb{R}^d$ .



**Example:** Consider the complete graph  $G = K_3$  with equal weights on edges. Embed  $y_1, y_2, y_3 \in \mathbb{R}^2$  in a unit circle at  $120^\circ$  apart. We get  $OPT_{SDP}(K_3) = \frac{1}{3} \cdot \frac{1}{2} \cdot 3(1 - \cos(120^\circ)) = \frac{1}{6} \cdot 3 \cdot \frac{3}{2} = \frac{3}{4}$  and  $OPT_{\max\text{-cut}}(K_3) = \frac{2}{3}$ . So we get approximation  $\frac{8}{9}$  which is better than the LP relaxation.

## 12.1.2 Rounding

Let  $z_u \in \mathbb{R}^n$  be an optimal solution to the SDP. To convert it into a cut, we need to pick sides.

### Theorem (Goemans/Williamson 1994)

Choose a random hyperplane through the origin of the unit sphere.

We choose a normal vector  $g \in \mathbb{R}^n$  from a Gaussian distribution and set  $x_u = \text{sign}(\langle g, z_u \rangle) = \text{sign}(g^T z_u)$  as the solution.

We can pick a random hyperplane through the origin in polynomial time by sampling a vector  $g = (g_1, \dots, g_n)$  by drawing  $g_i \in \mathcal{N}(0, 1)$ . If  $g'$  is the projection of  $g$  onto a two dimensional plane, then  $g' / \|g'\|_2$  is uniformly distributed over the unit circle in this plane.

### Analysis

The probability that the edge  $\{u, v\}$  crosses the cut is the same as the probability that  $z_u, z_v$  fall in different sides of the hyperplane, i.e.  $P[\{u, v\} \text{ crosses cut}] = P[g \text{ splits } z_u, z_v]$ .

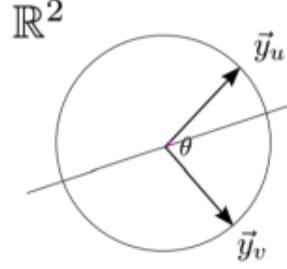
If we look at the problem in the plane:

The probability of splitting  $z_u, z_v$  is

$$P[\{u, v\} \text{ crosses cut}] = \frac{\theta}{\pi} = \frac{\cos^{-1}(z_u^T z_v)}{\pi} = \frac{\cos^{-1}(\gamma_{uv})}{\pi}$$

Let  $X$  be the random variable for value of the cut which is  $X = \sum_{\{u, v\} \in E} w_{uv} X_{uv}$ , and the

Figure 12.2: Plane of two vectors being cut by hyperplane.



expected value of the cut is

$$\mathbb{E}[\text{value of cut}] = \sum_{\{u,v\} \in E} w_{uv} \mathbb{E}[X_{uv}] = \sum_{\{u,v\} \in E} \frac{\cos^{-1}(\gamma_{uv})}{\pi}$$

Recall that

$$OPT_{SDP} = \sum_{\{u,v\} \in E} \frac{1}{2} w_{uv} (1 - z_u^T z_v) = \sum_{\{u,v\} \in E} \frac{1}{2} w_{uv} (1 - \gamma_{uv})$$

If we find  $\alpha$  such that

$$\frac{\cos^{-1}(\gamma_{uv})}{\pi} \geq \alpha \frac{1 - \gamma_{uv}}{2}, \quad \forall \gamma_{uv} \in [-1, 1]$$

Then we have an  $\alpha$ -approximation algorithm. For  $x \in [-1, 1]$ , we have

$$\frac{\cos^{-1}(x)}{\pi} \geq 0.878 \cdot \frac{1 - x}{2}$$

The **Random Hyperplane Cut algorithm** is a rounding procedure for SDPs if the constraints form a unit circle.

# **Part V**

## **Online Algorithms**

# Chapter 13

## Online Algorithms and Paging

### Definition: Competitive Analysis

Measures performance of an algorithm against the best algorithm that could see into the future.

It is a worst-case analysis.

Given an input sequence  $s = s_1, s_2, \dots, s_n$  of events. Let  $C_{opt}(s)$  be the minimum cost that any algorithm could achieve for  $s$ . Let  $C_A(s)$  be the cost of the online algorithm on  $s$ .

### Definition: Deterministic Competitive Ratio

A deterministic online algorithm  $A$  is  $k$ -competitive (has competitive ratio  $k$ ), if for all inputs  $s$ , we have

$$C_A(s) \leq kC_{opt}(s) + O(1)$$

### Definition: Randomized Competitive Ratio

A randomized online algorithm  $A$  is  $k$ -competitive (has competitive ratio  $k$ ), if for all inputs  $s$ , we have

$$\mathbb{E}[C_A(s)] \leq kC_{opt}(s)$$

## 13.1 Ski Rental Problem

### Ski Rental Problem

Decide whether to buy all the equipment or rent the equipment at the resort.

Suppose buying costs \$1000 dollars and renting at the resort cost \$100 per day.

Buying or renting depends on how many times you go skiing. If you go skiing 9 times or

less, then it is better to rent. If you go skiing at least 11 times, then it is better to buy. If you go exactly 10 times, then it does not matter.

This is an online algorithm because the algorithm decides when to buy, knowing only that we have gone skiing  $t$  times.

Any deterministic algorithm will rent  $t - 1$  times and buy on the  $t$ th time. Thus, the cost is  $100(t - 1) + 1000$ . If you go  $s < t$  times, then the cost is  $100s$  and the deterministic algorithm will buy at the beginning.

If  $t \leq 9$ , then rent and when  $t = 10$ , we buy. This is a 1.9-competitive algorithm.

Analysis: If  $t \leq 9$ , then the best strategy is to rent, so the cost is

$$\frac{C_A}{C_{opt}} = \frac{100t}{100t} = 1$$

If  $t \geq 10$ , we buy at the 10th time, so cost is

$$\frac{C_A}{C_{opt}} = \frac{100(9) + 1000}{1000} = 1.9$$

## 13.2 Dating Problem

### Dating Problem

There are  $n$  people you are interested in dating and would like to date the best person. Maximize the probability of dating the best person.

You could go out with all of them at the same time, but this is not possible. Thus, we have to go with one at a time. This is an online setting.

Consider the following algorithm:

1. Assume you ranked all the people from  $1, \dots, n$ .
2. Pick random order of  $n$  people:  $\pi$ .
3. Go out with  $n/e$  of the people and reject them.
4. After the first  $n/e$  dates, you will decide to settle if the person you found is better than anyone else you have dated before.

The algorithm picks the best person with probability  $\approx 1/e$ .

The more general algorithm is given a time  $t$ , go on  $t$  dates and from date  $t + 1$  onward, you decide to settle with a person who is better than the previous ones.

Suppose we pick a person at time  $k$ , then

$$P_k = P[\pi(k) = 1 \text{ and pick person at time } k]$$

Then our final success probability will be  $P = \sum_{k>t}^n P_k$ .

If  $\pi(k) = 1$ , then  $1 - P_k$  is the probability we picked a person between  $[t + 1, k - 1]$ , which means someone in this range is better than the first  $t$  people.

$$P_k = P[\pi(k) = 1 \cap \min\{\pi(1), \dots, \pi(k-1)\} \in \{\pi(1), \dots, \pi(t)\}] = \frac{(n-1)!}{n!} \cdot \frac{t}{k-1} = \frac{1}{n} \cdot \frac{t}{k-1}$$

We get

$$P = \sum_{k>t}^n \frac{1}{n} \cdot \frac{t}{k-1} = \frac{t}{n} \sum_{k>t}^n \frac{1}{k-1} \approx \frac{t}{n} \cdot (\ln n - \ln t) = \frac{t}{n} \ln \frac{n}{t}$$

Optimizing the above and we set  $t = n/e$ , which gives  $1/e$  probability.

To make it competitive

- Minimize the rank.
- The previous algorithm would either pick the best person or the last person, if  $\pi(k) = 1$  is within the first  $t$  people.
- With constant probability, the rank of the last person is  $\Omega(n)$ , which is within the bottom percentile.
- The expected rank is  $\Omega(n)$ .

This is not good. There is an algorithm that picks the person of average rank  $O(1)$ , so  $O(1)$ -competitive.

Based on computing time steps  $t_0 \leq t_1 \leq \dots$  and between time steps  $t_k$  and  $t_{k+1}$ , we will be willing to pick the person who is  $\leq k + 1$  best from the current seen list.

## 13.3 Online Paging Problem

Computer memory is hierarchical: cache  $\rightarrow$  L1  $\rightarrow$  L2  $\rightarrow$  main memory. The memory can be modelled in the following way:

- Each layer of memory is an array with a certain number of pages.
- A page stores the content of the item and its location in main memory.
- When we get a request, we first lookup in cache, then L1, then L2, then main memory.



- If the request is in the cache, we have a hit (which takes negligible time).
- Otherwise, we have a miss and need to fetch it in slower memory.
- In the negligible extra time, we copy the new data and location to the cache.
- If the cache is full, we must delete an old entry before copying in the new data and location.

### Paging Problem

Which entry in the cache do we delete when the cache is full?

The cost function is the number of cache misses. Assume that there is only cache and main memory.

### 13.3.1 Heuristics

#### Definition: Least Recently Used (LRU)

Delete page in cache whose most recent request happened furthest in the past.

#### Definition: Random

Delete random page.

#### Definition: First-In First-Out (FIFO)

Delete page that has been in cache the longest.

#### Definition: Least Frequently Used (LFU)

Delete page in cache which has requested least often.

- LRU is  $k$ -competitive.
- Random is  $k$ -competitive.
- FIFO is  $k$ -competitive.
- LFU is not competitive.

### Theorem

For a cache of size  $k$ , LRU is  $k$ -competitive.

Upper bound: divide input sequence into phases. The first phase starts immediately after the algorithm first faults and ends right after the algorithm faults  $k$  more times. Second phase starts after first phase and ends after  $k$  more faults.

We prove the optimal algorithm faults at least once per phase. This will give  $C_A \leq kC_{opt}$ .

**Proof.** Denote  $s^{(i)}$  be the  $i$ th phase. Then

$$C_{LRU}(s) = \sum_{i \geq 1} C_{LRU}(s^{(i)}) \leq \sum_{i \geq 1} kC_{opt}(s^{(i)}) = kC_{opt}(s)$$

Analysis: Same page faulted twice in one phase. Let  $s_1, \dots, s_t$  be a phase and  $p = s_1 = s_t$ . In order for page  $p$  to be faulted twice, then at least  $k - 1$  of the pages in cache have to be queried in  $s_2, \dots, s_{t-1}$ , then  $p$  would be least recently used so on  $s_t$ ,  $p$  faults. Thus, you would need to query at least  $k + 1$  distinct elements. By the pigeonhole principle, OPT must fault at least once.

Each page faulted once in a phase. Let  $s_1, \dots, s_t$  be a phase. Let page  $p$  comes before  $s_1$ . In the beginning of each phase, cache of OPT and cache of  $A$  must have  $p$ . Since OPT and  $A$  has a common page, then OPT must have faulted, so  $t = k$  distinct pages in the phase. Since there are  $k - 1$  spots (not  $p$ ) and we have  $k$  distinct pages, by pigeonhole principle, we have 1 fault.

#### Theorem

Any deterministic algorithm for paging with  $k$  pages is at least  $k$ -competitive.

**Proof.** Proof by trolling. Use  $k + 1$  pages and let  $A$  be paging algorithm. At each step, request a page that  $A$  does not have.  $A$  faults every single time.

The offline algorithm: on a cache miss, delete page which is requested furthest in the future. When the offline algorithm deletes a page, the next delete happens after at least  $k$  steps.

## Chapter 14

# Multiplicative Weights Update Method