CS466: Algorithm Design & Analysis

Felix Zhou

Spring 2020, University of Waterloo
from Professor Lap Chi Lau's Lectures

Contents

I Randomized Algorithms 19

Chapter 1

Introduction

In an introductory algorithms course, we learned about combinatorial techniques. These give rise to deterministic algorithms which always return optimal solutions.

Unfortunately, many combinatorial problems are NP-Complete. Even the fastest deterministic poly-time algorithms can be very complicated and difficult to implement.

We will focus on modern techniques in algorithm design using tools from probability theory, linear algebra, and continuous optimization.

1.1 Topics

Randomized Algorithms probabilistic inequalities, Markov chains, and Monte Carlo methods come to mind

Linear Algebra eigenvalues and eigenvectors and how they are useful

Linear Programming & Continuous Optimization this gives rise to approximation algorithms for NP-Hard problems

Each of the 3 topics can take an entire semester. We will take a broader view and ignore some applications or go into too much detail.

1.2 Maximum-Flow Minimum-Cut

Let $G = (V, E)$ be an unweighted, undirected graph with $s, t \in V$ be given.

Problem 1 (Minimum *st*-Cut)
Remove a minimum number of edges to disconnect s, t.

Problem 2 (Maximum *st*-Flow)
Find a maximum set of edge-disjoint st-paths.

Clearly, the maximum flow is bounded above by the minimum cut. The max-flow min-cut theorem says they are always equal.

Through the augmenting path algorihtm, we can achieve an $O(|V||E|)$ time algorithm to solve both.

With addition tools, a running time of $O(\min\{|E|^{1.5}, |E||V|^{\frac{2}{3}}\})$ is possible. This is a deterministic algorithm and can be extended to the weighted, directed case.

1.2.1 Techniques

Graph Sparsification any graph can be approximated by a sparse graph with cut values approximately the same

Linear Programming there is an integer programming formulation of maximum flow and therefore a linear programming relaxation. We will see a way to ”round” fractional optimal solutions to an integral one!

Spectral Techniques & Continuous Optimization graph sparsification and combinatorial ideas have been combined to give a nearly linear time algorithm for ”electrical flows”. We will see how this solver can be used to give a fast approximation of maximum flow

1.3 Quick Basic Probability Review

Sample Space Ω set of all outcomes, each with a probability associated with it

Event E subset of outcomes, $P(E) = \sum_{e \in E} P(e)$

Axioms
1. $0 \leq P(E) \leq 1$
2. $P(\Omega) = 1$
3. $P\left(\bigcup_{i=1} E_i\right) = \sum_i P(E_i)$ for disjoint E_i

Union Bound $P\left(\bigcup_i E_i\right) \leq \sum_i P(E_i)$

Inclusion-Exclusion Principle

Conditional Probability $P(E|F) = \frac{P(E \cap F)}{P(F)}$

Independence $P\left(\bigcap_i E_i\right) = \prod_i P(E_i)$

Total Probability $\{E_i\}$ a partition of Ω, $P(B) = \sum_i P(B \cap E_i) = \sum_i P(B|E_i)P(E_i)$

Baye's Rule $\{E_i\}$ a partition of Ω, $P(E_j|B) = \frac{P(B|E_j)P(E_j)}{\sum_i P(B|E_i)P(E_i)}$

Random Variable X is a function from $\Omega \to \mathbb{R}$ such that $P(X = a) = \sum_{x \in \Omega : X(s) = a} P(s)$

Independence X, Y are independent variables if and only if $P(X = x \cap Y = y) = P(X = x)P(Y = y)$

Expectation $E[x] := \sum_i iP(X = i)$

Linearity of Expectation $E\left[\sum_i X_i\right] = \sum_i E[X_i]$

Conditional Expectation $E[Y|Z = z] = \sum_y P(Y = y|Z = z)$ where $E[Y|Z]$ is a random variable of Z that takes on the value $E[Y|Z = z]$ if $Z = z$

Part I

Randomized Algorithms

Chapter 2

Randomized Minimum Cut

2.1 Problem

Problem 3 (Minimum Cut)
Given an unweighted, undirected graph $G = (V, E)$, find a minimum cardinality subset $F \subseteq E$ such that $G - F$ is disconnected.

Notice that for a fixed $s \in V$, s will belong to one of the two connected components of $G - F$. It suffices to compute the minimum s, t-cut for all possible t as the global minimum cut is also a minimum st-cut.

2.2 Randomized Algorithm

Karger gave a near linear time $\tilde{O}(|E|)$ algorithm.

The $\tilde{O}$ notation hides some poly-log factor in the run time.

We will present a $O(|V|^4)$-time algorithm and mention how it can be improved to $\tilde{O}(|V|)^2$.

2.3 Pseudocode

```
1) While there are more than two vertices
    a) pick a uniformly random edge and contract it
2) Output the remaining edges between vertices
```

Notice that each vertex in an intermediate graph induces a connected subgraph. So each cuz in an intermediate graph is a cut in the original graph. It follows that a min-cut in the intermediate graph is at least as large as a min-cut in the original graph.

2.4 Analysis

Theorem 2.4.1
The probability that the algorithm outputs a minimum cut is at least

$$\frac{2}{n(n-1)}$$

Proof
Let F be a minimum cut and let $k = |F|$.

If we never contract an edge in F until termination, then the algorithm succeeds. What is the chance that an edge in F is contracted in the i-th iteration?

The minimum-cut value in the i-th iteration is still at least k. Note this means every vertex has degree at least k. So by the handshake lemme, the number of edges in the i-iteration is at least

$$(n-i+1)\frac{k}{2}$$

Since we pick a random edge to contract, the chance that we pick an edge in F is at most

$$\frac{k}{\left[\frac{(n-i+1)k}{2}\right]}$$

So the probability that F survives is at least

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

2.4.1 Improving Success Probability

We can simply repeat the whole process many times. The failure probability after t repetitions is at most

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

Recall that $1 - x \leq e^{-x}$. This gives the failure probability of at most

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

2.4.2 Running Time

Now, one execution can be implemented in $O(n^2)$ time. It follows that the total time complexity is $O(n^4)$.

One observation is that the chance of choosing an edge from F is much higher near termination than at the first iteration. The Karger Stein Algorithm seeks to repeat later ierations but NOT the early iterations. This helps achieve the $\tilde{O}(n^2)$ running time mentioned earlier.

2.5 Corollaries & Generalizations

Corollary 2.5.0.1
There are at most $\binom{n}{2}$ minimum cuts in an undirected graph.

Proof
Each distinct minimum cut $S_1, \dots, S_k$ survives with probability $\frac{2}{n(n-1)}$.

The event that two different minimum cuts survive are disjoint since we cannot have two minimum cuts surviving at once. It follows that

$$1 \geq P\left(\bigcup_{i=1}^{k} S_i \text{ survives})\right) = k \cdot \frac{2}{n(n-1)}$$

This is non-trivial to prove using other arguments.

The algorithm can be extended to give a $n^{O(k)}$ time algorith for finding a minimum k-cut. This is NP-Hard when k is given as an input.

After studying graph sparsifiation, we may discuss Karger's near linear time minimum-cut algorithm.

Chapter 3

Tail Inequalities

Basic tail inequalities such as Markov, Chebyshev, and Chernoff are important tools in analyzing randomized algorithms.

3.1 Concentration Inequalities

On a high level, the goal is to give upper bounds on the probability that the value of a random variable is far from its expected value. In other words, randomized algorithms behane like what we expect with high probability.

3.1.1 Markov's Inequality

Theorem 3.1.1 (Markov's Inequality)
Let X be a non-negative discrete random variable. Then

$$P(X \geq a) \leq \frac{E[X]}{a}$$

for any $a > 0$.

Proof

$$
\begin{aligned}
E[X] &= \sum_i i \cdot P(X = i) \\
&\geq \sum_{i \geq a} i \cdot P(X = i) \\
&\geq \sum_{i \geq a} a \cdot P(X = i) \\
&= a \cdot P(X \geq a)
\end{aligned}
$$

3.1.2 Applications

Example 3.1.2 (Quicksort)
It is know that the expected runtime for randomized quicksort is $2n \ln n$. Markov's inequality tells that the probability that the run time exceeds

$$2cn \ln n$$

is at most $\frac{1}{c}$ for $c \geq 1$.

To see this, let $a = 2cn \ln n$.

Example 3.1.3 (Coin Flipping)
If we flip n fair coins, the expected number of heads is $\frac{n}{2}$.

Markov's inequality tells us that the probability that there are more than $\frac{3n}{4}$ heads is at most

$$\frac{2}{3}$$

Remark that Markov's inequality is most useful when all we now is the expected value. For the examples above, we can show much better bounds using Chernoff bounds as we have much more information.

Markov's bound can be tight. It does not hold for random variables in general. Moreover, no direct corollary allows us to bound $P(X \leq a)$.

3.2 Moments & Variance

Definition 3.2.1 (k-th Moment)
$E[X^k]$

Definition 3.2.2 (Variance)

$$\begin{aligned}\mathrm{Var}[X] &= E[(X - E[X])^2] \\ &= E[X^2 - 2XE[X] + E[X]^2] \\ &= E[X^2] - E[X]^2\end{aligned}$$

Definition 3.2.3 (Standard Deviation)
$\sigma[X] = \sqrt{\mathrm{Var}[X]}$

Definition 3.2.4 (Covariance)
$\mathrm{Cov}(X, Y) = E[(X - E[X])(Y - E[Y])]$

We say X, Y are positively (respectively negatively) correlated if

$$\mathrm{Cov}(X, Y) > 0, (< 0)$$

Proposition 3.2.1
$\mathrm{Var}[X + Y] = \mathrm{Var}[X] + \mathrm{Var}[Y] + 2\,\mathrm{Cov}(X, Y)$

Proposition 3.2.2
If X, Y are independent

$$\mathrm{Var}[X + Y] = \mathrm{Var}[X] + \mathrm{Var}[Y]$$

3.2.1 Chebyshev's Inequality

We would like to distinguish distributions that are concentrated around its expected value and those that are not.

Theorem 3.2.3 (Chebyshev's Inequality)
For any $a > 0$

$$P(|X - E[X]| \geq a) \leq \frac{\text{Var}[X]}{a^2}$$

Proof

$$\begin{aligned} P(|X - E[X]| \geq a) &= P((X - E[X])^2 \geq a^2) \\ &\leq \frac{E[(X - E[X])^2]}{a^2} && \text{Markov's Inequality} \\ &= \frac{\text{Var}[X]}{a^2} \end{aligned}$$

3.2.2 Examples

Example 3.2.4 (Coin Flipping)
Let X be the number of heads in n independent fair coin flips.

By independence, $\text{Var}[X] = \sum_{i=1}^n \text{Var}[X_i]$ where X_i is the indicator variable for the i-th coin flip. So

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

By Chebyshev's Inequality

$$\begin{aligned} P\left[X \geq \frac{3n}{4}\right] &\leq P\left(|X - E[X]| \geq \frac{n}{4}\right) \\ &\leq \frac{\text{Var}[X]}{\left(\frac{n}{4}\right)^2} \\ &= \frac{4}{n} \end{aligned}$$

Chebyshev's Inequality is useful when the second moment is easy to compute and suffices.

3.3 Sum of Independent Variables

The general objective is to bound

$$\begin{aligned} &P(X > (1+\epsilon)E[X]) && \text{upper tail} \\ &P(X < (1-\epsilon)E[X]) && \text{lower tail} \end{aligned}$$

We consider the situation when X is the sum of independent random variables, a common situation in the analysis of randomized algorithms.

The law of large number asserts that the sum of n independent identically distributed variables is approximately $n\mu$. To similar affect, the central limit theorem says that

$$\frac{X - n\mu}{\sqrt{n\sigma^2}} \to N(0,1)$$

The deviation from $n\mu$ is typically within $\sqrt{n}\sigma$, where σ denotes the standard deviation of a random variable.

Chernoff bounds give us quantitative estimates of the probabilities that X is far from $E[X]$ for any sufficiently large value of n.

3.3.1 Generalizing Markov's Inequality

For binomial variables with chance of "success" being p, we can simply compute

$$P(X \geq k) = \sum_{i \geq k} \binom{n}{i} p^i (1-p)^{n-i}$$

and show that it is very small when k is large. However, this does not generalize.

Instead, we extend the approach of using Markov's Inequality. Normally, Markov's Inequality is too weak, but as in the proof for Chebyshev's inequality, we may strengthen it by taking $2k$-th moment.

$$P(|X - E[X]| > a) = P((X - E[X])^{2k} > a^{2k}) \leq \frac{E[(X - E[X])^{2k}]}{a^{2k}}$$

We can consider

$$P(X \geq a) = P(e^{tX} \geq e^{ta}) \leq \frac{E[e^{tX}]}{e^{ta}}$$

for any $t > 0$.

The are at least two reasons to choose e^{tX} rather than some other increasing function.

Moments

Define

$$\begin{aligned} M_X(t) &= E[e^{tX}] \\ &= E\left[\sum_{i \geq 0} \frac{t^i}{i!} X^i\right] \\ &= \sum_{i \geq 0} \frac{t^i}{i!} E[X^i] \end{aligned}$$

If we have $M_X(t)$ and wish to compute $E[X^i]$, it suffices to compute

$$M_X^{(k)}(0)$$

the k-th derivative of $M_X(t)$ evaluated at $t = 0$.

$M_X(t)$ is the moment generating function giving us all information about moments. It gives a strong bound when applying Markov's inequality, as the denominator is exponentially large.

Independent Sum

If $X = X_1 + X_2$, two independent variables, then

$$E[e^{tX}] = E[e^{tX_1} e^{tX_2}] = E[e^{tX_1}]E[e^{tX_2}]$$

So this function is easy to compute when X is the sum of independent random variables.

3.4 Chernoff Bounds for Bounded Variables

Roughly speaking, Chernoff Bounds are obtained through Markov's inequality applied to the moment generating function as explained above. It is a general method rather than a specific inequality.

3.4.1 Heterogenous Coin Flips

Let $X_1, \dots, X_n$ be independent random variables with $X_i = 1$ with probability p_i and $X_i = 0$ otherwise.

Let $X := \sum_{i=1}^n X_i$ and

$$\mu = E[X] = \sum_{i=1}^n E[X_i] = \sum_{i=1}^n p_i$$

be the expected value.

We have

$$\begin{aligned}
E[e^{tX}] &= \prod_{i=1}^n E[e^{tX_i}] && \text{independence} \\
&= \prod_{i=1}^n \left(p_i e^{t\cdot 1} + (1-p_i)e^{t\cdot 0}\right) \\
&= \prod_{i=1}^n (1 + p_i(e^t - 1)) \\
&\leq \prod_{i=1}^n e^{p_i(e^t-1)} && 1 + x \leq e^x \\
&= e^{\sum_{i=1}^n p_i(e^t-1)} \\
&= e^{\mu(e^t-1)}
\end{aligned}$$

We can put in some specific parameters to get useful bounds.

Theorem 3.4.1
In the heterogenous coin flipping setting

1. for $\delta > 0$ we get
$$P(X \geq (1+\delta)\mu) < \left(\frac{e^\delta}{(1+\delta)^{1+\delta}}\right)^\mu$$
2. for $0 < \delta < 1$ we get
$$P(X \geq (1+\delta)\mu) < e^{-\frac{\delta^2\mu}{3}}$$
3. for $R \geq 6\mu$ we have
$$P(X \geq R) \leq 2^{-R}$$

Proof (1)
From our work before

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

The last function is minimized when $t = \ln(1+\delta)$, implying $P(X \geq (1+\delta)\mu) \leq \frac{e^{\mu\delta}}{(1+\delta)^{(1+\delta)\mu}}$.

Proof (2)
When $0 < \delta < 1$, it holds that $\frac{e^\delta}{(1+\delta)^{1+\delta}} \leq e^{-\frac{\delta^2}{3}}$.

Proof (3)
Let $R := (1+\delta)\mu$. When $R \geq 6\mu$, we have $\delta \geq 5$. Hence

$$\begin{aligned}
P(X \geq (1+\delta)\mu) &\leq \left(\frac{e^\delta}{(1+\delta)^{1+\delta}}\right)^\mu \\
&\leq \left(\frac{e}{1+\delta}\right)^{(1+\delta)\mu} \\
&\leq \left(\frac{e}{6}\right)^R \\
&\leq 2^{-R}
\end{aligned}$$

Similar bounds hold for the lower tail. A very similar proof results by setting $t < 0$.

Theorem 3.4.2
In the heterogenous coin flipping setting, we have for $0 < \delta < 1$

1. $P(X \leq (1-\delta)\mu) \leq \left(\frac{e^{-\delta}}{(1-\delta)^{1-\delta}}\right)^\mu$
2. $P(X \leq (1-\delta)\mu) \leq e^{\frac{-\mu\delta^2}{2}}$

Corollary 3.4.2.1
In the heterogenous coin flipping setting

$$P(|X-\mu| \geq \delta\mu) \leq 2e^{-\frac{\mu\delta^2}{3}}$$

for $0 < \delta < 1$.

Corollary 3.4.2.2 (Hoeffding Extension)
The same bounds hold when X_i is a random variable taking values in $[0,1]$ with mean p_i.

Proof (sketch)
e^{tX} is convex, and thus always lies below the straight line joining the endpoints $(0,1), (1,e^t)$. This line has the equation

$$y = \alpha x + \beta, \alpha = e^t - 1, \beta = 1$$

It follows that

$$\begin{aligned}E[e^{tX_i}] &\leq E[\alpha X_i + \beta] \\ &= \alpha p_i + \beta \\ &= 1 + p_i(e^t - 1)\end{aligned}$$

so the same calculations from above follow.

Remark that the same method holds for other random variables such as Poisson random variables, Gaussian random variables, etc.

It is often easier to compute the moments by computing the moment generating functions.

Chernoff bounds also hold fo rnegatively correlated variabels since then

$$E[e^{et(X+Y)}] \leq E[e^{tX}]E[e^{xY}]$$

For example, any two edges appearing in a random spanning tree are negatively correlated. It follows that Chernoff bounds still apply in the analysis of random spanning trees despite the dependency between variables.

3.4.2 Examples

Example 3.4.3 (Coin Flips)
Consider n independent fair coin flips, so the expected number fof heads is $\mu = \frac{n}{2}$.

By setting $\delta := \sqrt{\frac{60}{n}}$ we get

$$\begin{aligned}P(\text{number of heads - } \mu \geq \delta\mu) &\leq 2e^{-\frac{\mu\delta^2}{3}} \\ &= 2e^{-\frac{n\delta^2}{6}} \\ &= 2e^{-10}\end{aligned}$$

So with high probability, the number of heads is within $O(\sqrt{n})$ of the expected value.

On the other hand, we can further improve the bound from Chebychev to get

$$P\left(X \geq \frac{3n}{4}\right) \leq e^{-\frac{(n/2)(1/2)^2}{3}} = e^{-\frac{n}{24}}$$

which is exponentially smaller.

Example 3.4.4 (Probability Amplication)
The success probability of a randomized algorithm with one-sided error can be amplified easily. Say it is always correct when it says NO and correct with probability p when it says YES. To decrease failure probabiklity, we just repeat the algorithm k times or until it says NO.

The failure probability is at most $(1-p)^k$ when it says YES k times for a NO instance. For constant p repeating $\log n$ times decreases the failure to $O\left(\frac{1}{n}\right)$.

Suppose the randomized algorithm has two-sided errors. It has 60% chance of giving the correct answer but it could makes for both YES or NO output. We can run the algorithm k times and output the majority answer.

the majority answer is wrong when the randomized algorithm outputs NO for more than $\frac{k}{2}$ times. By Chernoff's bound, the majority answer being wrong has at most

$$P(\text{number of NOs} > (1+\frac{1}{4})E[\text{number of NOs}]) \leq e^{-\frac{\mu\delta^2}{3}} = e^{-0.4k\frac{(1/4)^2}{3}} = e^{-\frac{k}{120}}$$

By repeating $O(\log n)$ times, the failure probability is at most $O\left(\frac{1}{n}\right)$.

3.5 Applications of Tail Inequalities

3.5.1 Graph Sparsification

Consider an undirected graph $G = (V, E)$ with weight $w(e)$ on each edge $e \in E$. For $S \subseteq V$, let $\delta_G(S)$ be the set of edges with one endpoint in S and the other in $V - S$. Furthermore, write

$$w(\delta_G(S)) := \sum_{e \in \delta_G(S)} w(e)$$

to be the total weight of edges in $\delta_G(S)$.

We want to find a "sparse" graph which approximates the cut structures of G well.

Definition 3.5.1 (ϵ-Cut Approximator)
We say $H = (V, F)$ is an ϵ-cut approximator of $G = (V, E)$ if for all $S \subseteq V$

$$(1-\epsilon)w(\delta_E(S)) \leq w(\delta_H(S)) \leq (1+\epsilon)w(\delta_T(S))$$

Algorithm for Unweighted Inputs & Lower Bounded Cut Size

Let us assume the input $G = (V, E)$ is unweighted and has minimum-cut value $\Omega(\log|V|)$.

Set a sampling probability p. Fix an edge $e \in E(G)$. With probability p, put e in H with edge weight $w_e = \frac{1}{p}$.

The idea is to choose a p-fraction of edges and make their weight $\frac{1}{p}$, making the expected total weight of each cut in H the same as that in G. However, we need to ensure all cuts in H have approximately the same weight as that in G simultaneously so it is insufficient to simply consider expectation.

Theorem 3.5.1 (Karger)
Set

$$p = \frac{15 \ln n}{\epsilon^2 c}$$

where c denotes the minimum cut value of G.
Then H is an ϵ-cut approximator of G with $O(p|E(G)|)$ edges with probability at least

$$1 - \frac{4}{n}$$

Lemma 3.5.2
The number of cuts with at most αc edges for $\alpha \geq 1$ is at most

$$n^{2\alpha}$$

Proof
Consider a subset $S \subseteq V$. Say $\delta_G(S)$ has k edges. Notice that $k \geq c$ by definition.

By the linearity of expection

$$\begin{aligned} E\left[|\delta_H(S)|\right] &= \sum_{e \in \delta_G(S)} E[x_e] \qquad (*) \\ &= \sum_{e \in \delta_G(S)} (p \cdot 1 + (1-p) \cdot 0) \\ &= p|\delta_G(S)| \\ &= pk \end{aligned}$$

$(*)$ x_e is an indicator variable denoting if e is added to H or not.

Similarly

$$\begin{aligned} E[w(\delta_H(S))] &= \frac{1}{p} \cdot p|\delta_G(S)| \\ &= |\delta_G(S)| \\ &= k \end{aligned}$$

so the expected values are as desired.

Next, consider the probability that the actual value of $|\delta_G(S)|$ is "far" from expectation. Since $|\delta_H(S)|$ is a sum of independent indicator variables, we can apply Chernoff bound and get

$$\begin{aligned} P(||\delta_H(S)| - pk| \geq \epsilon pk) &\leq 2e^{-\frac{\epsilon^2 pk}{3}} \\ &= 2e^{-\frac{5k}{c}\ln n} && p = \frac{15\ln n}{\epsilon^2 c} \\ &= 2n^{-\frac{5k}{c}} \end{aligned}$$

Since $k \geq c$, the chance that $\delta_H(S)$ violates the requirements of an ϵ-cut approximator is at most

$$\frac{1}{n^5}$$

which is pretty small.

Now apply the lemma, as a naive union bound requires summing across exponentially many subsets of V which is undesirable.

$$\begin{aligned} P(\text{some cut is violated}) &\leq \sum_{S \subseteq V} P(\text{cut } S \text{ is violated}) && \text{union bound} \\ &= \sum_{S \subseteq V : 2^i c \leq |\delta_G(S)| \leq 2^{i+1}c, i \geq 0} P(\text{cut } S \text{ is violated}) \\ &\leq \sum_{\alpha = 2^i : i \geq 0} n^{4\alpha} P(\text{cut } S \text{ is violated} | \alpha c \leq |\delta_G(S)| \leq 2\alpha c) && \text{lemma} \\ &\leq \sum_{\alpha = 2^i : i \geq 0} n^{4\alpha} \cdot 2n^{-\frac{5\alpha c}{c}} && \text{Chernoff bound} \\ &= \sum_{\alpha = 2^i : i \geq 0} 2n^{-\alpha} \\ &\leq \frac{4}{n} && \text{geometric series} \end{aligned}$$

It follows that with probability at least $1 - \frac{4}{n}$, H is an ϵ-cut approximator of G.

Another simple application of Chernoff bound shows that H has $O(p \cdot |E(G)|)$ edges with high probability.

Although we did not explicitly use the lower bound for minimum cut, the statement is useless for

$$c <= 15n \ln n$$

as $p \geq 1$ and we essentially just return the original graph without any change.

Remark that the application of union bound and Chernoff bound is extremely powerful.

The result shows that for essentially complete graphs, there is an ϵ-cut approximator with

$$O\left(\frac{n\log n}{\epsilon^2}\right)$$

edges.

Appplications to Minimum Cut

We can sparsify a graph then compute the minimum st-cut. It can be shown that this is a

$$(1+3\epsilon) - \text{approximation}$$

but reduces run times depending on $|E| \in O(|V|^2)$.

Improvement

Benzczur and Karger designed a non-uniform sampling algorithm which each edge is sampled with probability inversely proportional to the "connectivity" of the two endpoints. Their result shows that an ϵ-cut approximator with $O\left(\frac{n\log n}{\epsilon^2}\right)$ edges is attained for any graph.

We will skip this result and instead return on this subject with spectral sparsification.

Minimum Cuts in Linear Time (Optional)

Let c be the minimum cut value of the the input graph $G=(V,E)$. Notice that G is then c-edge-connected.

Theorem 3.5.3
If G is c-edge-connected, then G has at least

$$\left\lfloor \frac{c}{2} \right\rfloor$$

edge-disjoint spanning trees.

Given all such spanning trees, we can find all one that crosses a minimum cut at most 2 times. This tree would be useful to compute the minimum cut S.

Sparsify the input, and compute $O(\log n)$ spanning trees. One such tree crosses a minimum cut at most two times.

3.5.2 Dimension Reduction (Optional)

Given n points in Euclidean space, we can always represent the vectors in n-dimensions. In general, we cannot do better is not distoration is allowed. Allowing some distortion allows us to significantly reduce the number of dimensions.

Theorem 3.5.4 (Johnson-Lindenstrauss Lemma)
Given any $\epsilon \in \left(0, \frac{1}{2}\right)$ and any set of points $X = \{X_1, X_2, \ldots, X_n\}$. There exists a map $A : X \to \mathbb{R}^k$ for $k = O\left(\frac{\log n}{\epsilon^2}\right)$ such that

$$1 - \epsilon \leq \frac{\|Ax_i - Ax_j\|_2^2}{\|x_i - x_j\|_2^2} \leq 1 + \epsilon$$

The Algorithm

We just project the points in a random k-dimensional subspace.

Let d be the dimension of original points. Let M be a $k \times d$ matrix, such that each entry of M is drawn from the normal $N(0,1)$ distribution. Define

$$Ax := \frac{1}{\sqrt{k}} M_x$$

(this is efficiently computable)

Since A is a linear transformation, the theorem can be reduced to the following.

Lemma 3.5.5
If A is constructed by the above algorithm with

$$k \in \Theta\left(\frac{1}{\epsilon^2} \log\left(\frac{1}{\delta}\right)\right)$$

then

$$P(1 - \epsilon \leq \|Ax\|_2^2 \leq 1 + \epsilon) \geq 1 - \delta$$

for any unit vector $x \in \mathbb{R}^d$ and any $\epsilon \in \left(0, \frac{1}{2}\right)$.

Given this lemma, we can set $\delta = \frac{1}{n^2}$ so for any pair $i \neq j$, the square length of $x_i - x_j$ is within $1 + \epsilon$ with probability at least $1 - \frac{1}{n^2}$. By union bound, the distances of all pairs are maintained within $1 + \epsilon$ with probability at least $\frac{1}{2}$.

Proof (Sketch)
Consider e_1. We want to find the length of the column Me_1. Notice that this follows a χ^2_k distribution and thus has expected length of Ae_1 is 1.

By setting k sufficiently harge, we can expect that the length is highly concentrated around its expectation.

With the $k \in O\left(\frac{\log n}{\epsilon^2}\right)$ described above, our intuition of Chernoff bound says the error probability is at most

$$2e^{-\frac{\mu\epsilon^2}{3}} \leq \frac{1}{n^2}$$

The actual proof requires us to handle two things.

(1) we cannot assume $x = e_1$

(2) the standard Chernoff bound does not apply since the random variables are unbounded.

Notice that for an arbitrary $y = Mx$ (x is unit vector), $y_j = \sum_{i=1}^d M_{ji}x_i$ where M_{ji} an $N(0,1)$ random variable. So y_j is a sum of Gaussian variables. In fact

$$y_j \sim N\left(0, \sum_{i=1}^d x_i^2\right) = N(0,1)$$

So we can just use the initial argument again.

By elementary calculus, we can compute the MGF of the independent Gaussians. Using this result, it is possible to show

$$P\left(\|Ax\|_2^2 \geq 1+\epsilon\right) \leq e^{-\frac{k\epsilon^2}{8}}$$

A lower bound comes from similar work.

By setting $k \in O\left(\frac{1}{\epsilon^2}\ln\left(\frac{1}{\delta}\right)\right)$, we get

$$P\left(|\|Ax\|_2^2 - 1| > \epsilon\right) \leq \delta$$

The result holds even when M is a random ±1 matrix. The proof is more difficult but the algorithm is much, much easier to implement.

Applications

An immediate and important use is to approximate the nearest neighbour search.

A linar scan takes $\Theta(n^2)$ time, but only $O(n \log n)$ after dimension reduction. However, this only works for Euclident distances only.

Another application is approximate matrix multiplication. we can dimension reduce the rows of A and columns of B so their product can be computed in $O(n^2 \log n)$ time.

Chapter 4

Balls & Bins

Balls and bins is a basic random process underlying several common phenomenons. There are applications to hashing.

4.1 Basic Results

We have m balls and n bins. Each ball is thrown to a uniformly random bin independently.

We would like to study what a typical endgame looks like.

4.1.1 Expected Number of Balls in a Bin

Let B_{ij} be the indicator variable that ball j is in the bin i. Then

$$\begin{aligned} E[\text{number of balls in bin } i] &= \sum j = 1^m E[B_{ij}] \\ &= \frac{m}{n} \end{aligned}$$

In particular, if $m = n$, the expected number of balls in a bin is one.

4.1.2 Expected Number of Empty Bins

Let Y_i be the indicator variable that bin i is empty. Then

$$\begin{aligned} E[Y_i] &= \left(1 - \frac{1}{n}\right)^m \\ &\approx e^{-\frac{m}{n}} \end{aligned}$$

So the expected number of empty bins is about

$$n \cdot e^{-\frac{m}{n}}$$

When $m = n$, we expect about $\frac{1}{e}$ of the bins are empty.

4.1.3 Maximum Load

What is the maximum number of balls which typically land in a bin? A simpler question is for which m do we expected to see two balls in a bin ("collision").

The birthday paradox is the case when $n = 365$. The probability that there are no collisions in the first m balls is

$$\begin{aligned} \prod_{i=1}^{m-1} \left(1 - \frac{i}{n}\right) &\leq e^{-\sum_{i=1}^{m-1} \frac{i}{n}} \\ &= e^{-\frac{m(m-1)}{2n}} \\ &\approx e^{-\frac{m^2}{2n}} \end{aligned}$$

This probability is smaller than $\frac{1}{2}$ when

$$m = \sqrt{2n \ln 2}$$

For $n = 365$, it says that when $m \geq 22.49$, the probability that the maximum load is at least two is at least $\frac{1}{2}$. This is very close to the exact answer.

to summarize, we expect to see a collision when

$$m = \Theta(\sqrt{n})$$

This observation is useful in different places.

An intuitive explanation is that there are m^2 pairs of possible collisions, so we expect some collision to occur when $m^2 \approx n$.

4.1.4 Maximum Load when $m = n$

The probability that a bin has at least k balls is at most

$$\binom{n}{k}\left(\frac{1}{n}\right)^k$$

by a union bound.

It is often that we have to deal with binomial coefficients. Some useful bounds are

$$\begin{aligned}\left(\frac{n}{k}\right)^k &< \binom{n}{k} \\ &< \frac{n^k}{k!} \\ &< \left(\frac{ne}{k}\right)^k\end{aligned}$$

Using this bound, the probability above is at most

$$\left(\frac{ne}{k}\right)^k\left(\frac{1}{n}\right)^k = \frac{e^k}{k^k}$$

By the union bound,

$$\begin{aligned}P(\text{some bin has at least } k \text{ balls}) &\leq n \cdot \frac{e^k}{k^k} \\ &= e^{\ln n + k - k\ln k}\end{aligned}$$

We would like to extimate the smallest k such that this probability is small enough. Rewording, we want the minimum k such that

$$k \ln k > \ln n$$

Setting $k = \frac{3\ln n}{\ln\ln n}$ would do. Thus with high probability, the maximum load is at most

$$O\left(\frac{\ln n}{\ln\ln n}\right)$$

4.1.5 Coupon Collector

For what m do we expect to have no empty bins? Leu X be the number of balls thrown until there are no empty bins. Let X_i be the number of balls thrown when there are exactly i bins.

So

$$E[X] = \sum_{i=1}^{n} E[X_i]$$

Notice that each X_i is a geometric random variable with parameter $p = \frac{i}{n}$. This is due to the fact that we look for the number of "failures" until our first "success" (choose empty bin).

The expected value of a geometric random variable with parameter p is

$$\frac{1}{p}$$

since

$$\begin{aligned}
E[X_i] &= \sum_{i\geq 1} iP(X_i = i) \\
&= \sum_{i\geq 1} P(X_i \geq i) \\
&= \sum_{i\geq 1} (1-p)^{i-1} \\
&= \frac{1}{1-(1-p)} \\
&= \frac{1}{p}
\end{aligned}$$

Thus

$$\begin{aligned}
E[X] &= \sum_{i=1}^{n} E[X_i] \\
&= \sum_{i=1}^{n} \frac{1}{i/n} \\
&= \sum_{i=1}^{n} \frac{n}{i} \\
&\approx n \ln n
\end{aligned}$$

This $n \ln n$ comes up as a lower bound for different things such as cover time of random walks in a complete graph, and the number of edges needed in graph sparsification by random samplying type algorithms.

4.2 Heuristic Arguments

We showed that the maximum load is

$$O\left(\frac{\ln n}{\ln \ln n}\right)$$

with high probability, but is it tight?

What is the probability of having an empty bin after throwing $n \ln n + cn$ balls? The problem with analyzing balls and bins is that the random variables are not independent. This means Chernoff bounds cannot be directly applied.

In this particular case, we observe that the events that two bins are nonempty are negatively correlated, and thus Chernoff bounds apply.

In the following, we pretend the variables are independent and come up with heuristic bounds. Later we mention these arguments can be made precise by Poisson Approximation.

4.2.1 Maximum Load

Let P_r be the probability that a bin has exactly r balls. Then

$$\begin{aligned} P_r &= \binom{m}{r} \left(\frac{1}{n}\right)^r \left(1 - \frac{1}{n}\right)^{m-r} \\ &= \frac{1}{r!} \frac{m(m-1)\dots(m-r+1)}{n^r} \left(1 - \frac{1}{n}\right)^{m-r} \end{aligned}$$

Assuming $m = n >> r$. Then

$$\begin{aligned} P_r &\approx \frac{1}{r!} \cdot 1 \cdot e^{-1} \\ &= \frac{1}{er!} \end{aligned}$$

We further assume that all bins are independent (not quite true, but not too far off). Then the probability that no bin has exactly r balls is at most

$$\left(1 - \frac{1}{er!}\right)^n \leq e^{-\frac{n}{er!}}$$

If this probability is very small, then with high probabiliy there will be some bin with at least r balls.

For $e^{-\frac{n}{er!}}$ to hold, it suffices to set

$$\begin{gathered}
-\frac{n}{er!} \leq -2\ln n \\
\iff \\
r! \leq \frac{n}{2e\ln n} \\
\iff \\
\ln r! \leq \ln n - \ln\ln n - \ln(2e)
\end{gathered} \tag{$*$}$$

$(*)$ By Stirling's approximation that

$$r! \leq e\sqrt{r}\left(\frac{r}{e}\right)^r \leq r\left(\frac{r}{e}\right)^r$$

we get

$$\begin{aligned}
\ln r! &= \sum_{i=1}^{r} \ln i \\
&\approx \int_1^r \ln x dx \\
&= x(\ln x - 1)\Big|_1^r \\
&= r\ln r - r \\
&\leq r\ln r - r + \ln r
\end{aligned}$$

Set $r = \frac{\ln n}{\ln\ln n}$. We get

$$\begin{aligned}
\ln r! &\leq \frac{\ln n}{\ln\ln n}\left(\ln\ln n - \ln\ln\ln n\right) - \frac{\ln n}{\ln\ln n} + \left(\ln\ln n - \ln\ln\ln n\right) \\
&\dots \\
&\leq \ln n - \ln\ln n - \ln(2e)
\end{aligned}$$

which shows that $(*)$ holds for the choice of r.

So there does indeed exist some bin with load

$$\Omega\left(\frac{\ln n}{\ln\ln n}\right)$$

with high probability.

4.2.2 Coupon Collector

To estimate the probability that some bin is empty after $n \ln n + cn$ balls, again we use

$$P_r = \frac{1}{r!}\frac{m(m-1)\dots(m-r+1)}{n^r}\left(1-\frac{1}{n}\right)^{m-r} \qquad \approx \frac{1}{r!}\cdot\left(\frac{m}{n}\right)^r \cdot e^{-\frac{m}{n}}$$

when $m, n >> r$.

For $m = n\ln n + cn$ we have $p_0 \approx \frac{e^{-c}}{n}$. So the probability of having some empty bin is

$$\begin{aligned} &\approx 1 - \left(1 - \frac{e^{-c}}{n}\right)^n \approx 1 - e^{-e^{-c}} \\ &\qquad\qquad = 1 - \frac{1}{e^{\frac{1}{e^c}}} \end{aligned}$$

When c is a large positive constant, this is very close to 0. On the other hand when c is a large negative constant, this is close to 1.

This is a "sharp" threshold phenomenon, for which we expect the even happens when there are very close to $n \ln n$ balls.

4.2.3 Poisson Approximation (Optional)

Notice that the approximation for P_r earlier is actually the probability distribution of a Poisson random variable.

Our work earlier actually assumes we are working with independent random variables. It can be shown that these heuristic arguments actually give some sort of bound on the balls and bins setting which differ by some constant factor.

4.3 Power of Two Choices (Optional)

Consider the variant where we choose 2 bins at random and put the ball in to the bin with lower number of balls.

We say a ball has height i if it is the i-th ball put into some bin. Let B_i be the number of balls with height i. Remark that

$$\frac{B_{i+1}}{n} \leq \left(\frac{B_i}{n}\right)^2$$

solving the recurrence with the deterministic base case

$$B_4 \leq \frac{n}{4}$$

gives that when

$$i = \log \log n + 3$$

$B_i \leq 1$.

So the maximum load is $O(\log \log n)$ under the power of two choices which is an exponential improvement!

The proof is not nearly detailed enough but can be made so with Chernoff bounds and some other steps.

Chapter 5

Hashing

Hashing is useful in designing efficient data structures for searching but also in data streaming algorithms and derandomization. One important concept is the notion of k-wise independent random variables.

5.1 Hash Functions

We want $O(1)$ time search using the RAM model. This means we can assume arbitrary position access is $O(1)$ and that the word size is sufficently large with word operations taking $O(1)$ time.

To store n members from a universe of M elements, we can easily do so with an array of size M. However, if $M >> n$ this would be impractical. We would like to use only an array of size $O(n)$ and still support searching in $O(1)$ time.

Definition 5.1.1 (Hash Function)
Used to map the elements of the bigger universe to the locations in the smaller table.

Definition 5.1.2 (Hash Table)
A data structure consisting of a table T with n cells indexed by from 0 to $n-1$ storing $O(\log m)$ bits each, and a hash function $h : M \to N$.

It is impossible for h to be injective if we do not know the keys in advance.

Definition 5.1.3 (Collision)
If $x \neq y$ but $h(x) = h(y)$.

The idea is to have a family of hash functions, so that the number of collisions is small with high probability. If we pick a random hash function from the family, we can assume the keys being stored at independent from the hash function we choose.

A natural setting is to simply consider the set of all functions from $M \to n$. This would bring us to the same balls-and-bins setting previously studied.

Suppose there are n keys. The expected number of keys in a location is 1, and the maximum load any single location has is

$$\Theta\left(\frac{\log n}{\log\log n}\right)$$

Using the idea of power of two choices, we can use two random functions h_1, h_2. When we insert a value x, we look at the locations $h_1(x), h_2(x)$ and store x in a least loaded location. When we search, we look at the linked lists in both locations $h_1(x), h_2(x)$.

Then we can reduce the maximum search time to

$$O(\log\log n)$$

while not increasing the average search time by more than a constant factor of 2.

5.1.1 Random Hash Functions

So far so good. However, we neglected the issue of computation time of $h(x)$ as well as the space requirement to store h. There is unfortunately no way to do it efficiently for truly random functions.

Consider a random function $h : M \to N$. Storing this table requires at least $m \log n$ bits. Each element in the image requires $\log n$ bits to remember its location.

Ideally, if we use $O(n)$ cells for the hash table with each cell storing $\log m$ bits, we would like the store the hash function using $O(1)$ cells. This means there is no overhead in storage requirement. Thus we can use at most $O(\log m)$ bits to represent the hash function, so our family has at most

$$\text{poly}(m)$$

functions instead of the n^m functions from $M \to N$.

Fortunately, choosing a hash function from a smaller family does not limit too much the properties guaranteed by random hash functions. However, we need a succint representation of a hash function to support fast query time and requires little storage space.

5.2 k-wise Independence

For a set of n independent random variables

$$P\left(\bigcap_{i=1} X_i = x_i\right) = \prod_{i=1}^{n} P(X_i = x_i)$$

k-wise independence is slightly weaker.

Definition 5.2.1 (k-wise Independent)
A set of random variables $X_1, \ldots, X_n$ is k-wise independent if for any subset $I \subseteq [n]$ with $|I| \leq k$ and any values of $x_i, i \in I$

$$P\left(\bigcap_{i\in I} X_i = x_i\right) = \prod_{i\in I} P(X_i = x_i)$$

Definition 5.2.2 (Pairwise Independence)
2-wise independence.

Example 5.2.1
Given random bits $X_1, \ldots, X_b$, we can generate $2^b - 1$ pairwise independent bits. For each $\varnothing \neq S \subseteq [b]$ let

$$Y_S := \oplus_{i\in S} X_i$$

where $\oplus$ denotes addition modulo 2.

Example 5.2.2
Given two independent uniformly random variables X_1, X_2 over $0, \ldots, p-1$, we can generate p pairwise independent random variables by setting

$$Y_i := (X_1 + iX_2) \mod p$$

for $i = 0, \ldots, p-1$ for some prime p.

The choice that p is prime is crucial.

5.2.1 Chebyshev's Inequality

We cannot apploy Chernoff bounds for pairwise independent random variables.

However, Chebyshev's Inequality still applies since for pairwise independent variables $X_1, \dots, X_n$

$$E[X_i X_j] = E[X_i]E[X_j]$$

for all $i \neq j$.

Thus covariance is still 0 for any $X_i, X_j, i \neq j$ so

$$\mathrm{Var}\left[\sum_{i=1}^{n} X_i\right] = \sum_{i=1}^{n} \mathrm{Var}[X_i]$$

and

$$P\left(|X - E[X]| \geq a\right) \leq \frac{\sum_{i=1}^{n} \mathrm{Var}[X_i]}{a^2}$$

5.3 Universal Hash Functions

Definition 5.3.1 (*k*-Universal)
Let U be a universe with $|U| \geq n$ and

$$V := \{0, \dots, n-1\}$$

A family of hash functions $\mathcal{H}$ from $U \to V$ is said to be k-universal if, for any distinct elements $x_1, \dots, x_k$ and a hash function h chosen uniformly random from $\mathcal{H}$

$$P_{h \in H}(h(x_1) = h(x_2) = \cdots = h(x_k)) \leq \frac{1}{n^{k-1}}$$

Definition 5.3.2 (Strongly *k*-Universal)
$\mathcal{H}$ is strongly k-universal if for any values $y_1, \dots, y_k \in \{0, \dots, n-1\}$ and a random hash function $h \in \mathcal{H}$

$$P_{h \in H}(h(x_1) = y_1, h(x_2) = y_2, \dots, h(x_k) = y_k) = \frac{1}{n^k}$$

for distinct $x_i \in U$.

We can think of $\mathcal{H}$ as strongly k-universal if the random variables

$$h(0), h(1), \dots, h(|U| - 1)$$

are k-wise independent when h is chosen uniformly random from $\mathcal{H}$.

With this connection, we will wee that the construction for generating k-wise independent random variables can be used to construct universal hash functions. We focus on 2-universal and strongly 2-universal hash functions.

5.3.1 2-Universal & Strongly 2-Universal Families of Hash Functions

Let us start with $U = V = \{0, \dots, p-1\}$ for some prime p. Let

$$h_{a,b}(x) := (ax + b) \mod p$$

and

$$\mathcal{H} := \{h_{a,b} : 0 \leq a, b \leq p-1\}$$

Proposition 5.3.1
$\mathcal{H}$ is strongly 2-universal.

Proof
We need to show

$$P_{a,b}\left((h_{a,b}(x_1) = y_1) \cap (h_{a,b}(x_2) = y_2)\right) = \frac{1}{p^2}$$

for any $y_1, y_2, x_1 \neq x_2$.

Assume the event above happens. Then

$$(ax_1 + b) \mod p = y_1, (ax_2 + b) \mod p = y_2$$

Given x_1, x_2, y_1, y_2, there are two linearly independent equations with two variables and thus a unique solution

$$a = (y_2 - y_1)(x_2 - x_1)^{-1} \mod p, b = (y_1 - ax_1) \mod p$$

Thus there is only one choice of a, b out of p^2 possibilities which satisfy the conditions as desired.

Now we look to extend the construction when $|U| >> |V|$. Indeed, let

$$U := \{0, 1, \dots, p^k - 1\}, V := \{0, 1, \dots, p-1\}$$

for some positive integer k and some prime p.

Interpret each $u \in U$ as a vector

$$\vec{u} = (u_0, \dots, u_{k-1})$$

where $0 \leq u_i \leq p-1$ and $\sum_{i=0}^{k-1} u_i p^i = u$. So $\vec{u}$ is the p-ary number where u_0 is the least significant digit and u_{k-1} is the most significant digit.

For any $\vec{a}, 0 \leq a_i \leq p-1$ and $0 \leq b \leq p-1$ let

$$h_{\vec{a},b}(\vec{u}) = \left(\sum_{i=0}^{p-1} a_i u_i + b\right) \mod p$$

and

$$\mathcal{H} := \{h_{\vec{a},b} : 0 \leq a_i \leq p-1, 0 \leq b \leq p-1\}$$

Proposition 5.3.2
$\mathcal{H}$ is strongly 2-universal.

Proof
We need to show that

$$P(h_{\vec{a},b}(\vec{u}) = y, h_{\vec{a},b}(\vec{w}) = \xi) = \frac{1}{p^2}$$

for any $y, \xi, \vec{u} \neq \vec{w}$.

Assume $u_{i_0} \neq w_{i_0}$. These conditions are equal to

$$a_{i_0}u_{i_0} + b = y - \left(\sum_{j \neq i_0} a_j u_j\right) \mod p, a_{i_0}w_{i_0} + b = \xi - \left(\sum_{j \neq i_0} b_j w_j\right) \mod p$$

which is again two systems of linearly independent equations with a single choice of (a_0, b) out of p^2 possibilities.

Proposition 5.3.3
With

$$h_{a,b}(x) = ((ax + b) \mod p) \mod n$$

the family

$$\mathcal{H} := \{h_{a,b} : 0 \leq a \leq p-1, 0 \leq b \leq p-1\}$$

is 2-universal.

Notice that there is a prime between $m, 2m$ for any $m \in \mathbb{Z}^+$. Thus the family above works for arbitrary m by choosing a $m \leq p \leq 2m$.

We can also define hash functions over other fields.

5.3.2 k-Universal Families

The idea is similar. Instead of generating a random one degree polynomial, we generate a random degree k poloynomial.

By polynomial interpolation, there is a unique degree k polynomial (up to scalar multiplication) with

$$p(x_i) = y_i$$

for any distinct points $x_1, \dots, x_k$ and values $y_1, \dots, y_k$.

It can be shown that with $h_{\vec{a}} : \{0, \dots, p-1\}^k \to \{0, \dots, n\}$ given by

$$h_{\vec{a}}(x) := \left(\sum_{i=0}^{k-1} a_k x^k \right) \mod p \quad \mod n$$

the family

$$\mathcal{H} := \{h_{\vec{a}} : 0 \leq a_i \leq p-1\}$$

is a k-universal hash family.

5.3.3 Hashing with 2-Universal Functions

Let $|V| = n$ and p a prime between $|U|, 2|U|$. With

$$h_{a,b}(x) = ((ax+b) \mod p) \mod n$$

the family

$$\mathcal{H} := \{h_{a,b} : 0 \leq a, b \leq p-1\}$$

is 2-universal.

Moreover, we can store this function with only 2 cells! The evaluation time of $h_{a,b}$ is also $O(1)$.

However, can they provide the same guarantees as random hash functions?

Expected Search Time

Lemma 5.3.4
Assume m elements $S \subseteq U$ are hashed into an n-bin hash table by using a randon hash function from a 2-universal family. For an arbitrary element x, let X be the number of elements at bin $h(x)$.
Then

$$E[X] \leq \begin{cases} \frac{m}{n}, & x \notin S \\ 1 + \frac{m-1}{n}, & x \in S \end{cases}$$

> **Proof**
> Let $X_i = 1$ if the i-th element in S is in the same bin as x and 0 otherwise. Since the hash function is chosen from a 2-universal family, it follows that
>
> $$P(X_i = 1) = \frac{1}{n}$$
>
> Therefore
>
> $$E[X] = \sum_{i=1}^{m} E[X_i] = \frac{m}{n}$$
>
> if $x \notin S$ and
>
> $$E[X] = E[X_j] + \sum_{i \neq j} E[X_i] = 1 + \frac{m-1}{n}$$
>
> if $x \in S$ is the j-th element of S.

Maximum Load

Unfortunately, we cannot guarantee the maximu load is still

$$O\left(\frac{\log n}{\log \log n}\right)$$

let $X_{ij} = 1$ if item i, j are mapped to the same bin and 0 elsewise. Then

$$X = \sum_{i,j} X_{ij}$$

is the number of collision pairs.

We have

$$\begin{aligned} E[X] &= \sum_{i,j} E[X_{ij}] \\ &= \sum_{i,j} P(h(x_i) = h(x_j)) \\ &\leq \sum_{i,j} \frac{1}{n} && \text{2-universality} \\ &= \binom{m}{2} \frac{1}{n} \\ &\leq \frac{m^2}{2n} \end{aligned}$$

By Markov's Inequality,

$$P\left(X \geq \frac{m^2}{n}\right) \leq \frac{1}{2}$$

or equivalently

$$P\left(X \leq \frac{m^2}{n}\right) \geq \frac{1}{2}$$

Suppose the maximum load is Y. Then there are at least $\binom{Y}{2}$ collision pairs.

Thus with probability at least $\frac{1}{2}$

$$\binom{Y}{2} \leq X \leq \frac{m^2}{n}$$

which implies

$$Y \leq m\sqrt{\frac{2}{n}} + 1$$

When $m = n$, the maximum load is

$$\sqrt{2n} + 1$$

with probability at least $\frac{1}{2}$.

We can guarantee the maximum load is

$$O\left(\frac{\log n}{\log\log n}\right)$$

with high probability at the cost of evaluation time

$$\Omega\left(\frac{\log n}{\log\log}\right)$$

which is not a good tradeoff.

5.4 Perfect Hashing

Given a fixed set S, we would like to build a data structure to support only search operations with excellent worst case guarantee.

Let $m = |S|$.

Definition 5.4.1 (Perfect Hash Function)
A hash function is perfect if it takes a constant number of word operations on $\log_2 m$-bit words so find an item or determine it does not exist.

Lemma 5.4.1
If $h \in \mathcal{H}$ ia a random hash function from a 2-universal family mapping the universe $U \to [0, n-1]$. then for any set S of size m when $m \leq \sqrt{n}$, the probability of h being perfect for S is at least

$$\frac{1}{2}$$

Proof
The expected number of collision pairs is less than

$$\frac{m^2}{2n}$$

By Markov's inequality, this implies

$$P\left(X \geq \frac{m^2}{n}\right) \leq \frac{1}{2}$$

When $n \geq m^2$, this means that it is perfect (no collision pair) with probability at least $\frac{1}{2}$.

To find a perfect hash function, we can generate random hash functions from $\mathcal{H}$ and check if it is perfect. On average we only need to check at most 2 hash functions. However, this scheme rquires

$$\Omega(m^2)$$

bins.

Two-Level Scheme

The idea is to first map the elements into a table of m bins/cells with maximum load $O(\sqrt{m})$. Then we build a secondary-level hash table for each bin. If a bin has k items, the second level hash table only needs $O(k^2)$ bins. Combining these will give a perfect hash function with only $O(m)$ bins.

Theorem 5.4.2
The two-level approach gives a perfect hashing scheme for m items using $O(m)$ bins.

> **Proof**
> As shown above, the nunber of collision pairs X in the first level is at most
>
> $$\frac{m^2}{n}$$
>
> with probability at least $\frac{1}{2}$.
>
> Thus for $m = n$, the number of collision pairs is at most m with probability $\frac{1}{2}$.
>
> The first level hash function can be found by trying and checking randon hash functions from a 2-universal family. On average, we only need to check at most 2 functions to find a first level hash function with at most m collision pairs.
>
> Let c_i be the number of items in the i-th bin. Then
>
> $$\text{number of collision paris} = \sum_{i=1}^{m} \binom{c_i}{2} \leq m$$
>
> We use a second-level hash function that gives no collisions using c_i^2 space for each bin with $c_i > 1$. By the above lemma, we can find a function by trying at most 2 randon hash functions on average. The total number of bins used is at most
>
> $$\begin{aligned} m + \sum_{i=1}^{m} c_i^2 &= m + 2\sum_{i=1}^{m} \binom{c_i}{2} + \sum_{i=1}^{m} c_i \\ &\leq m + 2m + m \\ &= 4m \end{aligned}$$

The extra space used to store the hash functions is at most $O(m)$ cells. Since there are at most $m + 1$ hash functions and each requires only $O(1)$ cells.

The search time is

$$O(1)$$

as it is $O(1)$ for each level.

Chapter 6

Data Streaming

6.1 Motivation

Suppose we have a massive data set where sublinear time and space is necessary. Here randomness is crucial as most tasks are impossible in the deterministic setting.

6.2 Heavy Hitters

Problem 4 (Heavy Hitters)
Given a data stream

$$x_1, \dots, x_T$$

where each

$$x_t = (i_t, c_t)$$

i_t is the ID of the t-th item and c_t is the weight associated with it.
Let

$$Q := \sum_{t=1}^{T} c_t$$

be the total weight.
For an ID i, let

$$C(i, T) = \sum_{1 \leq t \leq T : i_t = i} c_t$$

be the total weight coming from ID i.
We say ID i is a heavy hitter if

$$C(i, T) \geq q$$

for some threshold q.
Report all heavy hitters of this data stream.

Our goal is that ALL heavy hitters are reported. However, we allow false positives. Specifically, if

$$C(i, T) \leq q - \epsilon Q$$

then ID i is reported with probability at most δ.

Of course this is nontrivial only if $q > \epsilon Q$.

6.2.1 Hash Tables

We will use k independently chosen 2-universal hash functions

$$h_1, h_2, \dots, h_k$$

each mapping the universe into $[\ell]$.

We maintain a $k \times \ell$ table of counters, each counter $C_{a,j}$ adds the weight of items mapped to the j-th entry by the a-th hash function. Initially all counters are zero.

The algorithm is simple. Given $x_t = (i_t, c_t)$, increment

$$C_{a,h_a(i_t)}$$

by c_t.

After we read an entry, we report i_t if

$$\min_{j=h_a(i_t):1\leq a\leq k} C_{a,j} \geq q$$

Analysis

Clearly all heavy hitters are reported.

Suppose

$$C(i,T) \leq q - \epsilon Q$$

Then ID i is only reported if other IDs have contributed at least

$$\epsilon Q$$

to ALL its counters $C_{a,h_a(i)}$ for $a \in [k]$.

Let Z_a denote the total contributions of other IDs to the counter $C_{a,h_a(i)}$. Since h_a is chosen from a 2-universal family, the probability another item is mapped to $h_a(i)$ with probability at most

$$\frac{1}{\ell}$$

Thus

$$E[Z_a] \leq \frac{Q}{\ell}$$

By Markov's inequality

$$P(Z_a \geq \epsilon Q) \leq \frac{E[Z_a]}{\epsilon Q} \leq \frac{1}{\epsilon \ell}$$

So by independence

$$P(\min_a Z_a \geq \epsilon Q) \leq \left(\frac{1}{\epsilon \ell}\right)^k$$

as the hash functions are chosen independently.

Observe that by choosing

$$\ell := \frac{e}{\epsilon}, k := \ln\left(\frac{1}{\delta}\right)$$

this probability is at most δ.

The total space usage is

$$O\left(\frac{1}{\epsilon}\ln\frac{1}{\delta}\right) + O(k) = O\left(\frac{1}{\epsilon}\ln\frac{1}{\delta}\right)$$

(storing the hash functions).

The evaluation time is

$$O(k) = O\left(\ln\frac{1}{\delta}\right)$$

word operations to process a single element.

6.3 Distinct Elements

Problem 5 (Distinct Elements)
Given a stream $a_1, \dots, a_n$ where $a_i \in [m]$, output the number D of distinct elements in the data stream.

We will see a sublinear space algorithm which returns and estimate

$$(1-\epsilon)D \le Y \le (1+\epsilon)D$$

6.3.1 Strongly 2-Universal Family

We use a strongly 2-universal family to hash the input into a table of size m^3. The choice of m^3 is based on our previous observation that hasing m elements into m^2 locations, distinct elements have distinct hash values with probablity at least $\frac{1}{2}$. With the improvement to m^3, this happens with probability at least $1 - \frac{1}{2m}$.

Assuming hash values are evenly distributed, our naive intuition says that the t-th smallest hash VALUE is at

$$T \approx \frac{tm^3}{D}$$

which then gives an approximation of D.

The Algorithm

1) Choose a random hash function h from a strongly 2-universal family
2) For each a_i, compute $h(a_i)$. Then update our heap which stores the t smallest hash values seen so far
3) After all the data has been seen, let T be the t-th smallest hash value
4) return $Y := \frac{tm^3}{T}$

Analysis

Theorem 6.3.1
Set $t \in O\left(\frac{1}{\epsilon}\right)$, we have

$$(1-\epsilon)D \leq Y \leq (1+\epsilon)D$$

with probablity at least $\frac{2}{3}$.

Proof
Let us bound the probability that $Y = \frac{tm^3}{T} > (1+\epsilon)D$. In other words, assuming $\epsilon \leq 1$

$$T < \frac{tm^3}{(1+\epsilon)D} \leq \frac{tm^3\left(1-\frac{\epsilon}{2}\right)}{D}$$

Since T is the t-th smallest hash value, there are at least t hash values smaller than

$$\frac{tm^3\left(1-\frac{\epsilon}{2}\right)}{D}$$

Let $b_1, b_2, \dots, b_D$ be distinct elements in the data stream. Put X_i as the indicator variable when $h(b_i)$ is less than the value above. Then

$$\begin{aligned}
E[X] &= \sum_{i=1}^{D} E[X_i] \\
&= \sum_{i=1}^{D} P\left(X_i \leq \frac{tm^3(1-\frac{\epsilon}{2})}{D}\right) \\
&= \sum_{i=1}^{D} \frac{tm^3(1-\frac{\epsilon}{2})}{Dm^3} \\
&= t\left(1-\frac{\epsilon}{2}\right)
\end{aligned}$$

where X is the number of hash values smaller than our threshold.

We cannot apply Chernoff bounds since the X_i's are not necessarily independent. However, as h is from a strongly 2-universal family, each $X_i, X_j, i \neq j$ is pairwise independent. Thus variance is still linear and Chebyshev's inequality applies.

$$\begin{aligned}
\mathrm{Var}[X] &= \sum_{i=1}^{D} \mathrm{Var}[X_i] \\
&= \sum_{i=1}^{D} E[X_i](1 - E[X^i]) \\
&\leq \sum_{i=1}^{D} E[X_i] \\
&= t\left(1 - \frac{\epsilon}{2}\right) \\
&= E[X]
\end{aligned}$$

Thus

$$\begin{aligned}
P(Y > (1+\epsilon)D) &\leq P(X > t) \\
&= P\left(X > E[X] + \frac{t\epsilon}{2}\right) \\
&\leq P\left(|X - E[X]| > \frac{t\epsilon}{2}\right) \\
&\leq \frac{\mathrm{Var}[X]}{\left(\frac{t\epsilon}{2}\right)^2} \\
&\leq \frac{4t\left(1 - \frac{\epsilon}{2}\right)}{t^2\epsilon^2} \\
&\leq \frac{4}{t\epsilon^2}
\end{aligned}$$

By setting $t = \frac{24}{\epsilon^2}$

$$P(Y > (1+\epsilon)D) \leq \frac{1}{6}$$

The other inequality is identical. Putting everything together gives the result.

To boost the error probability, we can repeat

$$O\left(\log \frac{1}{\delta}\right)$$

many parallel copies of the algorithm and return the median.

This scheme uses a total of

$$O\left(\frac{1}{\epsilon^2} \log \frac{1}{\delta} \log m\right)$$

bits.

Now each heap update takes $O\left(\log \frac{1}{\epsilon}\right)$ operations as there are $O\left(\frac{1}{\epsilon^2}\right)$ numbers. Thus each input element takes

$$O\left(\log \frac{1}{\epsilon} \log 1\delta\right)$$

operations.

There is an optimal algorithm using

$$O\left(\frac{1}{\epsilon^2} + \log m\right)$$

space and $O(1)$ time.

6.4 Frequency Moments

Problem 6 (Frequency Moments)
Given a stream $a_1, \dots, a_n$ where $a_i \in [m]$, output

$$\sum_{i=1}^{m} x_i^p$$

where x_i is the number of items equal to i and p is a given number.

Observe that $p = 0$ is the distinct elements problem and $p = 1$ is just the total number of elements (trivial). What about $p = 2$?

6.4.1 Sketching for $p = 2$

The Algorithm

1. Let $r_1, \dots, r_m$ be independent random signs

$$P(r_i = 1) = P(r_i = -1) = \frac{1}{2}$$

2. Put $Y := \sum_{i=1}^{m} r_i x_i$
3. Return Y^2

First observe that we can keep a running some of Y so constant space is used.

Analysis

First we show that the expected value is what we want.

$$
\begin{aligned}
E[Y^2] &= E\left[\left(\sum_{i=1}^{m} r_i x_i\right)^2\right] \\
&= \sum_{i,j} E[r_i r_j x_i x_j] \\
&= \sum_{i,j} x_i x_j E[r_i r_j] \\
&= \sum_{i=1}^{m} x_i^2
\end{aligned}
$$

We wish to apply Chebyshev's inequality, this will require the second moment of Y^2.

$$
\begin{aligned}
E[Y^4] &= E\left[\left(\sum_{i=1}^{m} r_i x_i\right)^4\right] \\
&= \sum_{i,j,k,\ell} x_i x_j x_k x_\ell E[r_i r_j r_k r_\ell] \\
&= \sum_{i=1}^{m} x_i^4 + \binom{4}{2} \sum_{i<j} x_i^2 x_j^2 && (\star)
\end{aligned}
$$

$(\star)$ Observe that

$$
E[r_i r_j r_k r_\ell] = \begin{cases} 0, & \text{some index appear only once} \\ 1, & 1 = j = k = \ell \text{ or there are two pairs} \end{cases}
$$

It follows that

$$\begin{aligned}
\mathrm{Var}[Y^2] &= E[Y^4] - (E[Y^2])^2 \\
&= \sum_{i=1}^m x_i^4 + 6\sum_{i<j} x_i^2 x_j^2 - \left(\sum_{i=1}^m x_i^2\right)^2 \\
&= \sum_{i=1}^m x_i^4 + 6\sum_{i<j} x_i^2 x_j^2 - \sum_{i=1}^m x_i^4 - 2\sum_{i<j} x_i^2 x_j^2 \\
&= 4\sum_{i<j} x_i^2 x_j^2 \\
&\leq 2\left(\sum_{i=1}^m x_i^2\right)^2 \\
&= 2\left(E[Y^2]\right)^2
\end{aligned}$$

Recall that Chebyshev's inequality says

$$P(|Y^2 - E[Y^2]| \geq c\sqrt{\mathrm{Var}[Y^2]}) \leq \frac{1}{c^2}$$

thus

$$P(|Y^2 - E[Y^2]| \geq c\sqrt{2}E[Y^2]) \leq \frac{1}{c^2}$$

When $c = \sqrt{2}$ this is at most $\frac{1}{2}$.

To get a tighter bound, we want a random variable $\bar{Y}$ with the same expectation as Y^2 but smaller variance. The standard trick is to take

$$\bar{Y} := \frac{1}{k}\sum_{i=1}^k Y_i^2$$

which gives

$$\mathrm{Var}[\bar{Y}] = \frac{1}{k}\,\mathrm{Var}[Y^2] \leq \frac{1}{k}2\left(E[Y^2]\right)^2$$

Thus

$$\begin{aligned}
P\left(|\bar{Y} - E[\bar{Y}]| \geq c\sqrt{\frac{2}{k}}E[\bar{Y}]\right) &\leq \frac{1}{c^2} \\
&\Longrightarrow \\
P(|\bar{Y} - E[\bar{Y}]| \geq \epsilon E[Y^2]) &\leq \frac{1}{2}
\end{aligned}$$

when $c = \sqrt{2}, k = \frac{2}{\epsilon^2}$.

Space Requirement

We run $O\left(\frac{1}{\epsilon^2}\right)$ copies, requiring the same order of numbers.

However, we need to store the random bits! One trick is that we really only need 4-wise independence. Thus we can store $O(\log m)$ fully independent bits and produce m 4-wise independent bits.

This adds $O\left(\frac{1}{\epsilon^2}\log m\right)$ bits to our algorithm.

Remarks

This approach is called sketching.

It turns our polylogarithmic space is enough for $0 \leq p \leq 2$. However, it requires

$$\Theta\left(n^{1-\frac{2}{p}}\operatorname{polylog}(m)\right)$$

space for $p > 2$.

Chapter 7

Polynomial Identity Testing

We will explore simple algebraic ideas which have surprising application in designing fast and parallel algorithms.

7.1 String Equality

Problem 7 (String Equality)
Given two bit strings $a_1 a_2 \dots a_n$ and $b_1 b_2 \dots b_n$. Check whether the strings are the same by sending as few bits as possible.

This is easy if we send $n + 1$ bits. In fact, no deterministic algorithm can do better. This is provable using information theory.

7.1.1 Randomized Algorithm

The idea is to think of the strings as polynomials.

$$A(x) = \sum_{i=1}^{n} a_i x^i$$

$$B(x) = \sum_{j=1}^{n} b_j x^j$$

The Algorithm

1) Alice and Bob agree on a large prime p (agree on a finite field $\mathbb{F}$)

2) Alice picks some $r \in \mathbb{F}$ and send $r, A(r)$ to Bob

3) Bob computes $B(r)$ in $\mathbb{F}$ and return consistent if $A(r) = B(r)$, inconsistent otherwise

Notice that Alice sends $2\log_2 p$ bits and B sends 1 bit.

Analysis

First notice that there is only one-sided errors. We can only make mistakes where two strings are different but Bob returns consistent.

This event only happens when $A(x) \not\equiv B(x)$ but

$$A(r) = B(r) \iff (A - B)(r) = 0$$

By the Fundamental Theorem of Algebra, $A - B$ has at most n roots. Moreoever,

$$P(r \text{ is a root}) \leq \frac{n}{p}$$

By setting $p \geq 1000n$, then the probability of failure is at most

$$\frac{1}{1000}$$

by sending

$$O(\log_2 p) = O(\log_2 n)$$

bits.

Improving the Success Probability

We can repeat this k times to get a failure probability of at most

$$\left(\frac{1}{1000}\right)^k$$

where $p \geq 1000n$ by sending $O(k \log_2 n)$ bits.

A better way is to simply make $p \geq n^k$, so the failure probability of one run is reduced to

$$\frac{1}{n^{k-1}}$$

and we still only send $O(\log_2 p) = O(\log_2 n)$ bits.

7.2 Polynomial Identity Testing

This is one of the most fundamental problems that we do not know how to solve deterministically.

Problem 8 (Polynomial Identity Testing)
Given a multivariate polynomial

$$P(x_1, x_2, \ldots, x_n)$$

we want to determine if P is identically zero.

The problem is trivial if the polynomial is given explicitly. However, it becomes difficult if the polynomial is presented in a compact way.

7.2.1 Schwartz-Zippel Lemma

Theorem 7.2.1 (Schwartx-Zippel Lemma)
Let $Q(x) \in \mathbb{F}[x_1, \ldots, x_n]$ be of degree d.
Fix any finite set $S \subseteq \mathbb{F}$, and let $r_1, \ldots, r_n$ be independent uniform random elements of S.
Then

$$P[Q(r_1, \ldots, r_n) = 0] \leq \frac{d}{|S|}$$

if Q is not identically zero.

> **Proof**
> Induction on n.
>
> Factor out one variable x_1, and apply the induction hypothesis to two smaller polynomials to get a bound.

By choosing a sufficiently large p (ie $p \geq 2d$), there is a high probability that a non-zero polynomilal is still nonzero in the finite field of characteristic p.

we can substitute random values in

$$r_i \in \{0, 1, \ldots, p-1\}$$

and compute $Q(r)$.

If $Q \not\equiv 0$ then $Q(r) \neq 0$ with probability at least

$$\frac{1}{2}$$

7.3 Bipartite Matching

Problem 9 (Bipartite Matching)
Given a bipartite graph $G = (U \dot\cup W, E)$, find a maximum subset of vertex-disjoint edges.

Definition 7.3.1 (Perfect Matching)
When $|U| = |W| = n$, we say a matching $M \subseteq E$ is perfect if

$$|M| = n$$

7.3.1 Algebraic Formulation

Theorem 7.3.1 (Edmonds)
Let A be an $|U| \times |W|$ matrix of variables where

$$A_{ij} = x_{ij} \iff ij \in E$$

Then G has a perfect matching if and only if $\det A \not\equiv 0$.

Proof
By definition

$$\det A = \sum_{\sigma \in S_n} \operatorname{sgn}(\sigma) A_{q,\sigma(1)} A_{2,\sigma(2)} \dots A_{n,\sigma(n)}$$

Observe that any single term in the summation is non-zero if and only if there is a corresponding perfect matching.

So if there is no perfect matching, $\det A$ is identically zero.

Otherwise, we need to check that any non-zero monomial terms in the computation does not get cancelled out. This is clear from the fact that we choose different variables per edge.

7.3.2 Randomized Algebraic Algorithm

1) Choose a sufficiently large prime $p \in \Theta(\text{poly}(n))$, say $p \geq n^3$
2) Substitute each variable x_{ij} by a random value from $\{0, 1, \ldots, p-1\}$
3) Compute the numeric determinant using over the finite field of characteristic p
4) Return there is a perfect matching if and only if the determinant is not 0

Analysis

The determinant from bipartite matching is a multivariate polynomial with total degree at most n. If the graph has a perfect matching, then by Edmond's theorem the determinant is not identically 0. By the Schwartz-Zippel lemma, putting random values from the determinant results in a non-zero value with probability at least

$$1 - \frac{n}{p} \geq 1 - \frac{1}{n^2}$$

when $p \geq n^3$.

Complexity

Each number is at most p thus can be represented using $O(\log n)$ bits when p is a polynomial of n.

The determinant can be computed in $O(n^3)$ time with Gaussian Elimination. With fast matrix multiplication it is possible in $O(n^\omega) = O(n^{2.37})$ time.

This gives the fastest algorithm for bipartite matching in dense graphs.

It is also possible to return the set of edges in a perfect matching at the same time but it is far from trivial.

7.3.3 Algebraic Formulation for General Matching

Theorem 7.3.2 (Tutte)
Let A be an $n \times n$ matrix of variables where

$$A_{ij} = x_{ij}, A_{ji} = -x_{ij}$$

if $ij \in E$ and 0 otherwise.
Then G has a perfect matching if and only if

$$\det A \not\equiv 0$$

Proof
The idea is that terms of the determinant correspond to a cycle cover of G. Moreoever, any odd cycle would be "cancelled" out and so the remaining graph consists only of even cycles.

7.4 Parallel Algorithm for Algebraic Problems

One feature of this algebraic approach is that it leads to parallel algorithms. This is because algebraic problems can be solved efficiently in parallel using a divide-and-conquer approach.

Many combinatorial optimization problems have an algebraic formulation.

Theorem 7.4.1
The determinant of an $n \times n$ matrix can be computed in

$$O(\log^2 n)$$

time using $O(n^{\omega+1})$ processors.

7.4.1 Isolation Lemma

How can we "focus" all our processors on finding the SAME matching.

Lemma 7.4.2 (Isolation)
Given any family of subsets on a groud set of n elements, if we assign an independent uniform random weight from $[2n]$ to each element, there is a unique minimum weight subset with probability at least

$$\frac{1}{2}$$

Proof
Fix an element v. Let F_v be the family of sets containing v, and $F_{\bar{v}}$ the family of sets not containing v.

Define

$$\alpha_v := \min_{S \in F_{\bar{v}}} w(S) - \min_{R \in F_v} w(R - v)$$

and observe that the quantity is independent of the weight $w(v)$.

Case I: $w(v) > \alpha_v$ v is NOT in any minimum weight set as the minimum weight set not containing v already beats any set containing v.

Case II: $w(v) < \alpha_v$ Then v is in EVERY minimum weight set as the minimum weight set contaning v beats any which does not.

Case III: $w(v) = \alpha_v$ Here v is ambiguous.

Observe that since α_v is independent of $w(v)$

$$P(v \text{ is ambiguous}) = \frac{1}{2n}$$

since only one of $2n$ possible values of $w(v)$ attain this.

But union bound,

$$P(\text{some element is ambiguous}) \leq \frac{1}{2}$$

So with probability at least $\frac{1}{2}$, there are no ambiguous elements and all Case I elements must be in the minimum weight set.

7.4.2 Isolation Lemma for Matchings

Apply the isolation lemma with the ground set being the edge set and the set of perfect matchings being the family of subsets. Then there is a unique minimum weight perfect matching!

Edges in Minimum Weight Perfect Matching

Suppose we know the unique minimum weight perfect matching M has weight W.

Proposition 7.4.3
$e = uv \in M$ if and only if

$$G - u - v$$

has a unique minimum weight perfect matching with weight $W - w_{uv}$.

Thus assuming we know how to find the weight of the minimum weight perfect matching, we are done.

Weight of Minimum Weight Perfect Matching

We return to the Tutte matrix and encode the weight of the edge $w(e)$ into the variable x_e

$$x_e = 2^{w(e)}$$

If W is the unique minimum weight among matchings, then the determinant is in the form

$$\det A = 2^W \pm 2 \cdot 2^W (L), L \in \mathbb{Z}$$

Observe that

$$\frac{\det A}{2^k} = \begin{cases} \text{even}, & k < W \\ \text{not divisible}, & k > W \\ \text{odd}, & k = W \end{cases}$$

So we can just binary search all possible values of $k \leq 2n^2$ in parallel.

It is important that the determinant can be computed efficiently, as each number has at most $2n$-bits. This is why the isolation lemma is crucial. It shows we can isolate a minimum weight perfect matching with relatively small weights.

Chapter 8

Network Coding

Recall that the Schwartz-Zippel lemma says that a low degree non-zero multivariate polynomial evaluated at random points have only a small chance of being zero.

8.1 Network Multicasting

Problem 10 (Network Multicasting)
Given a directed acyclic graph $G = (V, E)$, a source $s \in V$, and a set of receiver vertices $\{t_1, \dots, t_\ell\} \subseteq V$, send data to all receivers simultaneously while maximizing the transmission rate.

Assume that the capacity of each arc is 1. The classical method of attack is to find the maximum number of edge-disjoint steiner trees which span $s, t_1, \dots, t_\ell$. This is known to be NP-hard.

Intuitively, it is relatively easy to send data to a single receiver, since that is equivalent to finding the maximum *st*-flow for a fixed t. However, it is difficult to maximize transmission rate among all receivers.

8.2 Network Coding

The idea is that information is different from commodity flow. We can employ encoding and decoding to improve transmission rate.

We allow arithmetic operations over a field. Data send on an edge is an arbitrary function of its predecessors.

Network coding can in fact achieve the optimal rate for multicasting!

Theorem 8.2.1 (Max-Information-Flow Min-Cut Theorem)
If the source has at least k edge-disjoint paths to each receiver, the source can send k units of data to all receivers simultaneously using network coding.

We interpret this as being able to send k units to every single receiver individually is equivalent to being able to send k units to all receivers.

This is optimal, since we cannot hope to send more than k units as there is a cut of k-edges.

8.2.1 Linear Network Coding

We can actually restrict the output functions to linear functions of the input. It is important to use different linear combinations to prevent information loss.

Decoding simplifies to solving a system of k unknowns using k equations.

8.2.2 Polynomial Time Algorithms

Deterministic Algorithm

Polynomial time algorithms are designed for finding the optimal scheme. However, schemes are centralized and difficult to implement in practice.

Randomized Algorithm

While allowing for some mistakes, we are able to achieve a completely decentralized scheme achieving the optimal rate.

8.2.3 Randomized Scheme

Our goal is to attain k target transmission rate, assuming there are k edge-disjoint paths between st_i for each i. But the st_i, st_j-paths are not necessarily disjoint between i, j!

By creating a super source node or super receiver nodes, we can assume without loss of generality that the source has exactly k outgoing edges (none incoming) and each receiver has exactly k incoming edges (none outgoing).

The source wants to send k messages

$$x_1, x_2, \dots, x_k$$

to the receivers.

The Algorithm

1) Choose a sufficiently large prime field $\mathbb{F}_p$
2) The source s sends $x_1, x_2, \dots, x_k$ on its outgoing edges, with x_i on its i-th edge
3) Follow a topological ordering of $v_1, \dots, v_n$ of the DAG
    a) For each vertex v_i, call the incoming messages $m_1, \dots, m_\ell$
    b) For each outgoing edge e of v_i, send a random linear combination $m_1, \dots, m_\ell$ along each edge

Global & Local Coefficients

Each edge is sending a linear combination of the ORIGINAL source messages. Call the vector representing this combination the global encoding vector.

At each node, we make the decision to take some random linear combination of the inputs. Call the vector representing this linear combination the local encoding coefficients.

Once the local encoding coefficients are fixed, the global encoding vectors are determined.

Decodability

For a receiver t with k incoming vectors, let m_i be the i-th incoming message and let the global encoding vector of the i-th incoming edge be

$$(c_{i_1}, \dots, v_{ik})$$

In vector notation

$$m = Cx$$

where x is the vector of k messages.

This has a unique solution if and only if C is invertible or $\det C = 0$.

Analysis

We argue that the probability of $\det C = 0$ is small. The idea is to show that $\det C$ is a low degree polynomial of the local encoding coefficients.

Lemma 8.2.2
For vertex v_i in the topological ordering, for each outgoing edge e of v_i, each entry c_j of its global encoding vector is a multivariate polynomial of the local encoding coefficients (perhaps of edges not incident to v_i) of total degree at most $i-1$.

Proof
This holds for $i = 1$ trivially.

Let $\vec{c}_j$ denote the incoming (input) global encoding vectors. Then fix an edge e. c_e is a a linear combination of $\vec{c}_j$'s with coefficients independently randomly chosen.

By induction each entry $\vec{c}_j$ is a polynomial of degree at most $i-2$. Thus the degree of each entry in $\vec{c}_e$ is at most $i-1$.

By the lemma, each entry in a receiver matrix C is a multivariate polynomial of the local encoding coefficients of total degree at most n. Thus the determinant is a polynomial of the local encoding coefficients of degree at most kn.

Schwartz-Zippel says if the determinant is non-zero, then the probability that the determinant is zero is at most

$$\frac{kn}{|\mathbb{F}|}$$

By choosing $|F| \in \Theta(kn^3)$, this is at most $\frac{1}{n^2}$. By a union bound the probability that some receiver fails is at most

$$\frac{1}{n}$$

It remains to prove the assumption that the polynomial is not identically zero.

Proposition 8.2.3
If there are k edge-disjoint paths to a receiver, then the receiver matrix C satisfies

$$\det C \not\equiv 0$$

Proof
It suffices to choose some local encoding vectors so that $\det C$ does not evaluate to zero. In other words, find inputs such that the polynomial is non-zero, then it cannot be identically zero.

> Simply forward the k-bits down through the k-edge disjoint paths so at the end $C = I_k$ which certainly has $\det C \not\equiv 0$ and we are done.

8.2.4 Efficient Encoding

In a node with in-degree and out degree d, the encoding can be accomplished in $O(kd^2)$ time per node.

We would like to reduce the in and out degrees of intermediate nodes.

Superconcentrator

Definition 8.2.1 (Superconcentrator)
A DAG with d input nodes and d output nodes.
In addition, there are d internal nodes and each internal node has constant in-degree and constant out-degree.
Finally, for any $1 \leq k \leq d$, for any subset X of k input modes and any subset of Y of k output nodes, there are k vertex disjoint paths between X, Y.

Observe that for our purposes, it is similar to a complete bipartite graph, but has only $O(d)$ edges overall!

This is an efficient object to reduce the in-degree of a vertex while maintaining connectivity.

We can replace vertices with in-degree d by a superconcentrator with d inputs ($O(d)$ vertices). This the vectors of the outgoing edges of a single vertex can be computed in $O(kd)$ time.

This is optimal since reading the incoming edges would take $\Omega(kd)$ time.

Chapter 9

Probabilistic Methods

Loosely speaking, probabilistic methods use techniques from probability to prove mathematical statements which otherwise has no relation to probability.

One example we have already seen is the bound on the number of minimum cuts given by Karger's algorithm.

9.1 First Moment Method

The general idea is to compute $E[X]$ and argue there is an outcome with $X \geq E[X]$ or $X \leq E[X]$. For a non-negative random variable X, Markov's inequality implies that

$$P(X \geq 1) \leq E[X]$$

Thus when X is integral and $E[X] << 1$, $X = 0$ with high probability.

9.1.1 Ramsey Graphs

Can we find a 2-edge-coloring of K_n such that there are no "large" monochromatic cliques?

Theorem 9.1.1
If

$$\binom{n}{k} 2^{-\binom{k}{2}+1} < 1$$

then it is possible to color the edges of the complete graph on n vertices so that there are no monochromatic cliques of size k.

Proof
Color each edge red or blue independently with probability $\frac{1}{2}$. Let $S \subseteq V$ and define X_S to be the indicator variable which is 1 if S is monochromatic. Furthermore, put X as the random variable indicating the number of monochromatic cliques of size k.

$$\begin{aligned} E[X_S] &= P(X_S = 1) \\ &= 2 \cdot 2^{-\binom{k}{2}} \\ &= 2^{-\binom{k}{2}+1} \\ E[X] &= \sum_{S \subseteq V : |S| = k} E[X_S] \\ &= \binom{n}{k} 2^{-\binom{n}{k}+1} \\ &< 1 \end{aligned}$$

Thus there must be at least one outcome where there are no monochromatic subgraphs of size k.

This means if $k > 2\log_2 n$, then a random coloring will work with high probability. There is no deterministic polynomial time algorithm to construct such a coloring. Even a coloring with size $\sqrt{n}$ requires extremely complex tools.

Corollary 9.1.1.1
If we generate a random graph where each pair of vertices are adjacent with probability $\frac{1}{2}$, then with high probability, there are no cliques of size more than $2\log_2 n$ and no independent sets of size more than $2\log_2 n$.

9.1.2 Magical Graphs

Definition 9.1.1 ((n, m, d)-Magical Graph)
Bipartite $G = (U, W; E)$ is called an (n, m, d)-magical graph if

(i) $|U| = n, |W| = m$

(ii) every vertex in U has degree d

(iii) $|N(S)| > |S|$ for every $S \subseteq U$ such that $|S| \leq \frac{|U|}{2}$

Theorem 9.1.2 (Hall)
Given a bipartite graph $G = (U, W; E)$ and $S \subseteq U$, S can be perfectly matched to W if and only if

$$|N(X)| \geq |X|$$

for any subset $X \subseteq S$.

Proposition 9.1.3
A magical graph is a sparse bipartite graph $G = (U, W; E)$ with the property that every subset $S \subseteq U$ with

$$|S| \leq \frac{|U|}{2}$$

has a perfect matching in to W.

Theorem 9.1.4
For every n, $m \geq \frac{3n}{4}$, and $d \geq 8$, there exists an (n, m, d)-magical graph.

Proof
Let G be a random graph where there are n vertices on the left and m vertices on the right. Each left vertex is connected to d random vertices on the right independently.

For

$$|S| \leq \frac{|U|}{2}, |T| = |S|$$

Let $X_{S,T}$ be the indicator randon variable that all edges from S go to T.

Put

$$t := |T|, s := |S|$$

and

$$X := \sum_{S \subseteq U : |S| \leq \frac{|U|}{2}} \sum_{T \subseteq W : |T| = |S|} X_{S,T}$$

$$
\begin{aligned}
E[X] &= \sum_{S\subseteq U:|S|\leq\frac{|U|}{2}} \sum_{T\subseteq W:|T|=|S|} E[X_{S,T}] \\
&= \sum_{S\subseteq U:|S|\leq\frac{|U|}{2}} \sum_{T\subseteq W:|T|=|S|} \left(\frac{t}{m}\right)^{ds} \\
&= \sum_{1\leq s\leq\frac{n}{2}} \binom{n}{s}\binom{m}{t}\left(\frac{t}{m}\right)^{ds} \\
&= \sum_{1\leq s\leq\frac{n}{2}} \binom{n}{s}\binom{m}{s}\left(\frac{s}{m}\right)^{ds} && t = s \\
&\leq \sum_{1\leq s\leq\frac{n}{2}} \left(\frac{ne}{s}\cdot\frac{me}{s}\cdot\left(\frac{s}{m}\right)^{d}\right)^{s} \\
&\leq \sum_{1\leq s\leq\frac{n}{2}} \left(\frac{4}{3}e^2\left(\frac{s}{m}\right)^{d-2}\right)^{s} && m \geq \frac{3n}{4} \\
&\leq \sum_{1\leq s\leq\frac{n}{2}} \left(\frac{4}{3}e^2\left(\frac{2}{3}\right)^{d-2}\right)^{s} && s \leq \frac{n}{2}, m \geq \frac{3n}{4} \\
&\leq \sum_{1\leq s\leq\frac{n}{2}} \left(\frac{1}{2}\right)^{s} && d \geq 8 \\
&< 1
\end{aligned}
$$

9.1.3 Superconcentrators

Recall that a superconcentrator is a directed acyclic graph with n input nodes I and n output nodes O which satisfies the following property: For any $1 \leq k \leq n$ and any subsets $S \subseteq I, T \subseteq O$ with $|S| = |T| = k$, there are k vertex disjoint paths between S and T.

Valiant conjectured no super concentrator exists with $O(n)$ edges.

Recursive Construction

The base case is where $n = O(1)$. We can simply use a complete bipartite graph.

Assume a superconcentrator C exists with $\frac{3n}{4}$ inputs and outputs. Take G_1, G_2 to be two $\left(n, \frac{3n}{4}, O(1)\right)$-magical graphs. Attach $G_1 - C - G_2$ together so that there are n inputs and n outputs from G_1, G_2. Moreover, add n additional edges forming a perfecting matching between the inputs and outputs.

Now, any input and output vertices connected by a matching edge has edge disjoint paths.

Observe that ignoring such vertices, there can be at most $\frac{n}{2}$ input and output vertices remaining! Then we can use the magical graphs G_1, G_2 to get a perfect matching from the inputs to the C and C to outputs. Then we use the fact that C was a superconcentrator and get at most $\frac{n}{2}$ edge disjoint paths.

Analysis

Let $E(n)$ denote the number edges in our construction of n input and output nodes.

$$\begin{aligned} E(n) &= 2dn + n + E\left(\frac{3n}{4}\right) \\ &= E\left(\frac{3n}{4}\right) + O(n) \\ &\in O(n) \end{aligned}$$

9.2 Second Moment Method

We may also wish to apply some concentration inequality to show that $P(X \geq 1)$ is large (X is integral).

Lemma 9.2.1
We have

$$P(X = 0) \leq \frac{\mathrm{Var}[X]}{E[X]^2}$$

which shows that $P(X \geq 1)$ is large.

Proof
By Chebychev's inequality

$$\begin{aligned} P(X = 0) &\leq P(|X - E[X]| \geq E[X]) \\ &\leq \frac{\mathrm{Var}[X]}{E[X]^2} \end{aligned}$$

Corollary 9.2.1.1
If $\mathrm{Var}[X] \in o(E[X]^2)$ or equivalently $E[X^2] = (1 + o(1))E[X]^2$ then

$$P(X > 0)$$

with high probability.

9.2.1 Threshold Behavior in Random Graphs

Let $G_{n,p}$ be a random graph on n vertices, where each pair of vertices are adjacent with probability p independently.

Definition 9.2.1 (Threshold Behavior)
A property has a threshold behavior if there is a function $f(n)$ such that

$$\lim_{n} \frac{g(n)}{f(n)} = 0$$

implies $G_{n,g(n)}$ does NOT satisfy the property with high probability. But when

$$\lim \frac{h(n)}{g(n)} = \infty$$

then $G_{n,h(n)}$ satisfies the property with high probability.

Definition 9.2.2 (Sharp Threshold Behavior)
A property has a threshold behavior if there is a function $f(n)$ such that for any $\epsilon > 0$

$$G_{n,(1-\epsilon)f(n)}$$

does not satisfy the property with high probability. But

$$G_{n,(1+\epsilon)f(n)}$$

does in fact satisfy the property with high probability.

9.2.2 Cliques of Size 4

Theorem 9.2.2
The property of having a clique of size 4 has a threshold function

$$f(n) = n^{-\frac{2}{3}}$$

Proof
Let X_S be the indicator variable which says $S \subseteq V$ is a clique of size 4 and define

$$X := \sum_{S \subseteq V : |S| = 4} X_S$$

We have

$$\begin{aligned} E[X] &= \sum_{S \subseteq V : |S| = 4} E[X_S] \\ &= \binom{n}{4} p^6 \end{aligned}$$

If we choose $p \in o(n^{-\frac{2}{3}})$ this expectation approaches 0.

On the other hand $p \in \omega(n^{-\frac{2}{3}})$ then $E[X] \to \infty$. We need however to show that

$$E[X^2] = (1 + o(1)) E[X]^2$$

This can be done by considering

$$E[X^2] = \sum_S \sum_T E[X_S X_T]$$

which can be computed by considering the different sizes of $S \cap T$.

9.2.3 Diameter 2

Theorem 9.2.3
The property of having diameter 2 has sharp threshold

$$p = \sqrt{\frac{2 \ln n}{n}}$$

Proof
Call a prit of vertices i, j a bad pair if there is not edge between i, j and no other vertex in G is adjacent to both i, j.

Observe that a graph is of diameter 2 if and only if there is no bad pair.

> Let $X_{i,j}$ be the indicator variable such that vertices i, j form a bad pair. We have
>
> $$\begin{aligned} E[X] &= \sum_{i<j} E[X_{i,j}] \\ &= \binom{n}{2}(1-p)(1-p^2)^{n-2} \end{aligned}$$
>
> We can show by computation that for $c > 2$ and
>
> $$p = \sqrt{\frac{c \ln n}{n}}$$
>
> then $E[X] \to 0$.
>
> On the other hand we can show $E[X^2] = (1 + o(1))E[X]^2$ when
>
> $$p = \sqrt{\frac{c \ln n}{n}}$$
>
> for $c < 2$.

9.3 Local Lemma

9.3.1 Finding a Good Outcome

Let $E_1, \dots, E_n$ be a set of bad events. A typical goal with probabilistic methods is to show that there is an outcome which is not in ANY of the bad events.

$$P\left(\bigcap_{i=1}^{n} E_i^c\right) > 0$$

Two situation arise where this is easy.

If the E_i's are mututally independent, then

$$P(\cap E_i^c) = \prod P(E_i^c) > 0$$

Also if the sum of probability of bad events is less than 1, applies a union bound suffices.

9.3.2 Local Union Bound

Definition 9.3.1 (Mutually Independent)
We say and event E is mututally independent of a set of events $E_1, \dots, E_m$ if

$$P\left(E | \bigcap_{i\in I} E_i\right) = P(E)$$

So $P(E)$ does not change conditioned on any subsets of $E_1, \dots, E_m$.

Lemma 9.3.1 (Lovász Local Lemma)
Let $E_1, \dots, E_n$ be a set of events and the following holds

1. $P(E_i) \leq p$ for $1 \leq i \leq n$
2. Every event is mutually independent of all but at most d other events
3. $4dp \leq 1$

Then

$$P\left(\bigcap_{i=1}^{n} E_i^c\right) > 0$$

We sometimes refer to d as the maximum degree of the "dependency graph".

Non-Constructive Proof

The original proof by Lovász is non-constructive. It is an inductive proof where the probabiliy of a good event can be exponentially small.

We will instead delay to see an algorithmic proof.

9.3.3 k-SAT

Problem 11 (k-SAT)
Given a Boolean formula where each clause has exactly k literals, the goal is to find a satisfying assignment to the variables when it exists.

The problem in general is NP-hard, but we can use the local lemma to prove that a formula always has a satisfying assignment when it is "under constraint".

Theorem 9.3.2
If no variables in a k-SAT formula appears in more than

$$T = \frac{2^k}{4k}$$

clauses, then the formula has a satisfying assignment.

Proof
Produce a random assignment where each variable has $\frac{1}{2}$ chance of being true of false. Let E_i be the event that the i-th clause is violated.

First remark that

$$p := P(E_i) = \frac{1}{2^k}$$

as everything in the clause is not true.

Secondly, if two clauses do not share variables, then they are independent. But each variable appears in at most T clauses, and there are at most k distinct variables per clause

$$d \leq k \cdot T \leq 2^{k-2}$$

Finally

$$4pd = 4\frac{1}{2^k}2^{k-2} = 1$$

thus by the local lemma there is at least one event outcome which avoids all E_i's

This bound is essentially tight!

9.3.4 Edge-Disjoint Paths

Theorem 9.3.3
For each i, let $\mathscr{P}_i$ be a set of L paths connecting s_i and t_i. Suppose each path in $\mathscr{P}_i$ does NOT share edges with more than C paths in $\mathscr{P}_j$ for $i \neq j$. Moreoever, suppose $\frac{8kC}{L} \leq 1$, then there is $P_i \in \mathscr{P}_i$ so that

$$\{P_1, \ldots, P_k\}$$

are edge-disjoint.

Proof
For each i, pick a random path independently from $\mathscr{P}_i$ with probability $\frac{1}{L}$. Let E_{ij} be the event that P_i, P_j are NOT edge-disjoint.

Clearly

$$p := P(E_{ij}) \leq \frac{C}{L}$$

Moreoever, for a fixed E_{ij}, it is only dependent on E_{ia} for $a \in [k]$ and E_{bj} for $b \in [k]$, so

$$d \leq 2k$$

Finally by assumption

$$4pd = \frac{8kC}{L} \leq 1$$

so the local lemma applies and at least one outcome is desirable.

9.3.5 Algorithmic Implications for k-SAT

1) Fix an ordering $C_1, \dots, C_m$
2) Find a random assignment of the variables
3) for $1 \leq i \leq m$, if C_i is not satisfied, FIX(C_i)

FIX is the following subroutine

1) Substitute the variables in C_i with random variables
2) While there is a clause D that shares variables with C and D is NOT satsified, FIX(D)

Analysis

We argue that if the algorithm does not terminate fast enough, we can leverage it to compress random bits.

Proposition 9.3.4
A random string of k bits can be compressed into $k - c$ bits with probability at most

$$2^{-c}$$

for any $c \geq 1$.

Suppose the algorithm runs for t steps but has not terminated. We used n initial random bits to fix a clause. Each subsequence call to FIX requires k extra random bits.

All in all, we consumed

$$n + tk$$

random bits up to now.

Encoding

We encode the execution tree in order to recover the random bits.

1. We use a 0-bit and $\log_2 m$ bits to represent going from the root to a clause
2. We use a 0-bit and $\log_2 d$ bits to represent going from the current clause ot a neighbour clause
3. We use a 1-bit to represent going up.
4. We use n bits to encode the final assignment

We only descend from the root at most m times. the edge which represents descending (and the ascending) from (and to) the root is used at most

$$m(\log_2 m + 2)$$

times.

For each non-root node edge, this similarly happens

$$t(\log_2 d + 2)$$

times.

Finally, we use n bits for the final assignment.

Suppose we can recover the $n+tk$ random bits consumed during the duration of the algorithm, then with proability at least $1 - 2^{-c}$,

$$n(\log_2 m + 2) + t(\log_2 d + 2) + n \geq n + tk - c$$

This can be arranged to give

$$t \leq \frac{m(\log_2 m + c)}{k - \log_2 d - 2} \in O(m \log m)$$

given that $k - \log_2 d - 2 \geq \epsilon$, which is equivalent to

$$d \leq 2^{k-2-\epsilon}$$

This is only slightly stronger than the assumption made in the local lemma!

Decoding

Let the random bits used in the initial algorithm be

$$v_1, \ldots, v_n$$

Then let the injected bits be

$$r_1^i, r_2^i, \ldots, r_k^i$$

for each $1 \leq i \leq t$.

Start with variables v_j's representing the initial assignment.

Whenever we find a clause to fix, we learn k bits since there is only ONE assignment of those k variables which violates that clause.

Then we replace the corresponding variables which tracks unknown bits with the values we learned. Then we replace them with variables representing the newly consumed k bits introduced at the step of the fix.

When we arrive at the final assignment, all the rest of the variables are recovered.

Chapter 10

Random Walks

10.1 Random Walks

Definition 10.1.1 (Random Walk)
Given some graph G and a starting vertex s, iteratively move from the current vertex to a uniformly random neighbour of the current vertex.
This is a simple random process.

10.1.1 Basic Questions

Definition 10.1.2 (Stationary Distribution)
Suppose as $t \to \infty$, the probability distribution of being at a vertex converges.
The stationary distribution is this distribution.

Is there a limiting distribution? If so, what does it look like?

Definition 10.1.3 (Mixing Time)
Suppose a limiting distirbution exists.
The mixing time is the time it takes to converge.

If the mixing time is small, there are applications to random sampling.

Definition 10.1.4 (Hitting Time)
Starting from a vertex s, this is the expected nunber of steps to first reach a vertex t.

Definition 10.1.5 (Cover Time)
The time it takes to reach every vertex of the graph at least once.

Approaches

One way of approaching the first two questions is by “coupling” probability distributions.

We will attack then through linear algebra, by looking at the eigenvalues of the transition matrix.

10.2 Markov Chains

Let $p_t(i)$ be the probability distribution of being at a vertex i at time t. Then for all $0 \leq k \leq n-1$

$$p_{t+1}(j) = \sum_{i=1}^{n-1} p_t(i) \cdot P_{i,j}$$

where P is the $n \times n$ transition matrix of non-negative real number whose columns sum to 1.

More succintly

$$p_{t+1} = p_t P = p_0 P^t$$

where $p_t \in \mathbb{R}^n$ represents the probability distribution among vertices at time t.

Definition 10.2.1 (Markov Chain)
A Markov chain is a mathematical system that experiences transitions from one state to another according to certain probabilistic rules.
The defining characteristic of a Markov chain is that no matter how the process arrived at its present state, the possible future states are fixed.

10.2.1 Irreducible Markov Chains

Definition 10.2.2 (Irreducible)
A Markov Chain is irreducible if the underlying directed graph is strongly connected.

If the underlying graph is not strongly connected, we can always decompose them into strongly connected components, between which form a DAG.

By topologically sorting the DAG, we can argue that eventually the walk converges within one of the sink components.

Thus if there is only one such sink, we have reduced the problem to the irreducible case. On the other hand, if there are multiple sinks, the limiting distribution depends on the starting point since we can end up in different sinks depending on the starting point.

10.2.2 Aperiodic Markov Chains

Definition 10.2.3 (Period)
The period of state i is

$$\text{period}(i) := \gcd\{t : P_{i,i}^t > 0\}$$

Definition 10.2.4 (Aperiodic)
A state is aperiodic if

$$\text{period}(i) = 1$$

Definition 10.2.5 (Aperiodic Markov Chain)
A Markov chain is aperiodic if all state are aperiodic.

Definition 10.2.6 (Periodic Markov Chain)
If it is not aperiodic.

Lemma 10.2.1
For any finite, irrreducible, and aperiodic Markov chain, there is some $T < \infty$ such that

$$(P^t)_{ij} > 0$$

for all i, j and $t \geq T$.

10.2.3 Stationary Distribution

Definition 10.2.7 (Stationary)
A probability distribution π is stationary if

$$\pi = \pi P$$

We need some way of defining distance and convergence

Definition 10.2.8 (Total Variation Distance)
Given p, q, the total variation distance if

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

Definition 10.2.9 (Converge)
We say $p_t \to q$ if

$$\lim_t d(p_t, q) = 0$$

10.2.4 The Fundamental Theorem of Markov Chains

Definition 10.2.10 (Return Time)
The return time from $i \to i$ is

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

The expected return time is

$$h_i := E[H_i]$$

Theorem 10.2.2 (Fundamental Theorem of Markov Chains)
For any finite, irreducible, aperiodic Markov chain, the following hold.

1. There is a stationary distribution π
2. The distribution p_t converges to π as $t \to \infty$, regardless of the distribution p_0
3. There is a unique stationary distribution
4. $\pi(i) = \frac{1}{h_i}$

10.3 Pagerank

Given a collection of websites, creat a directed graph where $p_1 p_2$ is an arc if and only if p_1 has a link to p_2.

We would like to know which web pages are the most relevant based on the number of incoming links.

10.3.1 The Algorithm

1. Initially, each page has pagerank value $\frac{1}{n}$
2. In each step, each page divides its pagerank value uniformly to its outgoing links and sends these equal shares to the pages that it points to
3. Each page updates its new pagerank value to be the sun of the shares it receives
4. Repeat until the pagerank values converge

10.3.2 Analysis

Proposition 10.3.1
The equilibrium pagerank values are the probabilities in the stationary distributions.

Proof
Fundamental theorem of Markov Chains.

This shows the pagerank values are a function of the graph structure, and not the initial values.

10.3.3 In Practice

The graph may not be irreducible and eperiodic.

We can instead fix a tiny $\epsilon > 0$, and only give out $1 - \epsilon$ of the pagerank values to neighbours. The extra ϵ is spread among the value to all remaining vertices.

The idea is to go to a neighbour with probability $1 - \epsilon$ and with probability ϵ, go to a random vertex.

This is equivalent to augmenting the graph until it is complete with very small weighted edges. Clearly the new graph is irreducible and aperiodic, hence the pagerank values are unique.

10.4 Perfect Matching in Regular Bipartite Graphs

A well-known consequence of Hall's theorem that every regular bipartite graph has a perfect matching.

We can find a perfect matching in $O(m)$ time. Assume that the degree is a power of 2, The idea is to find an Eulerian orientation and throw away half the edges to get a $\frac{d}{2}$-regular bipartite graph. Repeat until the graph is 1-regular, this is a perfect matching.

We will show a randomized algorithm to find a perfect matching in $O(n \log n)$ time, which is sublinear when the graph is dense!

10.4.1 Traditional Approach

A direct algorithm is to repeatedly find an augmenting path to enlarge the current matching.

Theorem 10.4.1
A matching is maximum if and only if there is no augmenting path.

Augmenting Path Algorithm

1) Start with the empty matching
2) While there is an augmenting path, use the path to enlarge the matching ($O(m)$ time)
3) Return the matching

The loop occurs at most n times and each iteration requires some graph exploration which is $O(m)$ time. Thus the overall complexity is

$$O(mn)$$

10.4.2 Random Walk

The idea is to replace BFS/DFS by a random walk.

Let A, B be a bipartition with $|A| = |B|$. add an unmatched A-univeral node s and an unmatched B-universal node t.

Direct the edges from s to A and B to t. Within the original graph, direct the matching edges towards A, and the remaining edges towards B.

Proposition 10.4.2
Let G_1 be the original undirected graph and G_2 be the directed graph obtained from G_1 from the description above.
G_1 has an augmenting path if and only if G_2 has a directed path from s to t.

Now take G_2 and create G_3 as follows. Contract all matched edges. Moreoever, for every unmatched vertex in A, add 0 or more edges from s until it has indegree equal to outdegree.

Similarly for unmatched vertices of B, add 0 or more edges to t until it has indegree equal to outdegree. Finally, add edges from t to s until both t and s have indegree equal to outdegree.

Proposition 10.4.3
G_2 has a s, t-dipath if and only if G_3 has a (non-trivial) s, s-dipath.

Analysis

The expected time to find an augmenting path is equal to the expected return time h_s in G_3.

Thus the expected time to find an augmenting path is equal to

$$\frac{1}{\pi(s)}$$

in G_3.

Proposition 10.4.4
G_3 is an Eulerian digraph.

Proof
This holds for unmatched vertices and s, t simply by construction. For every identified node x_{ab} which contracted the edge $ab \in E(G_1)$, the degree of a, b was both d by regularity. The indegree of x_{ab} is precisely then $d - 1$ and so is its outdegree.

Lemma 10.4.5
The stationary distribution of an Eulerian digraph is

$$\pi(i) = \frac{\text{outdeg}(i)}{m}$$

for all $i \in V$.

Proof

We have

$$\begin{aligned}(\pi P)_j &= \sum_{ij\in E} \pi(i) P_{i,j} \\ &= \sum_{ij\in E} \frac{\text{outdeg}(i)}{m} \cdot \frac{1}{\text{outdeg}(i)} \\ &= \frac{\text{indeg}(j)}{m} \\ &= \frac{\text{outdeg}(j)}{m} \\ &= \pi(j)\end{aligned}$$

Corollary 10.4.5.1

The expected time to find an augmenting path is

$$\frac{1}{\pi(j)} = \frac{m}{\text{outdeg}(s)}$$

Time Complexity

In the i-th iteration when there are only i edges in the matching,

$$\text{outdeg}(s) = (n-i)d$$

Remark that

$$|E(G_3)| \leq 4|E(G_1)| = 4dn$$

so the expected time to find an augmenting path in the i-th iteration is

$$\frac{m}{\text{outdeg}(s)} \leq \frac{4n}{n-i}$$

It follows that the total expected runnning time is

$$\sum_{i=1}^{n} \frac{4n}{n-i} \in O(n \log n)$$

Implementation

We do not need to actually construct G_3. With the appropriate data structures, it is still possible to implement the algorithm in

$$O(n \log n)$$

time.

Part II

Spectral Analysis

Chapter 11

Spectral Graph Theory

11.1 Linear Algebra Review

Definition 11.1.1 (Eigenvector)
$0 \neq v \in V$ such that

$$Av = \lambda v$$

for non-zero λ.

Definition 11.1.2 (Eigenvalue)
λ.

Notice we do not require $\lambda \neq 0$. This is because a matrix can be singular. thus it would have eigenvalue 0. The converse also holds.

Definition 11.1.3 (Characteristic Polynomial)
Let x be a variable in $\mathbb{F}$

$$\det(A - xI)$$

is the characteristic polynomial.

The roots of the characteristic polynomial are the eigenvalues of A.

Definition 11.1.4 (Algebraic Multiplicity)
The maximum k such that

$$(x-\lambda)^k$$

is a factor the characteristic polynomial.

Definition 11.1.5 (Geometric Multiplicity)
The dimension of the eigenspace.

In general the algebraic multiplicity dominates the geometric multiplicity.

11.1.1 Real Symmetric Matrices

Theorem 11.1.1 (Spectral Theorem)
Let $A \in \mathbb{R}^{n\times n}$ be real and symmetric.
There is an orthonormal basis of eigenvectors and all eigenvalues are real numbers.

Proof
We know A has at least one eigenvalue and thus one eigenvector by the Fundamental Theorem of Algebra. Let v_1, λ_1 be the eigenvector and eigenvalue, respectively.

We claim λ_1 is real. Indeed

$$\begin{aligned}\lambda_1 v_1^T \bar{v}_1 &= v^T A^T \bar{v}_1\\ &= v^T A \bar{v}_1\\ &= v^T(\overline{\lambda v_1})\\ &= \bar{\lambda} v^T \bar{v}\end{aligned}$$

so $\lambda = \bar{\lambda}$ implies that the imaginary component of λ is 0 and so $\lambda \in \mathbb{R}$.

Extend v_1 to a orthonormal basis of $\mathbb{R}^n$.

$$v_1, w_2, \dots, w_n$$

Then let W be the span of $w_2, \dots, w_n$.

> Choose $w \in W$.
>
> $$\begin{aligned} \langle Aw, v_1 \rangle &= \langle w, A^* v_1 \rangle \\ &= \langle w, A v_1 \rangle \\ &= \langle w, \lambda v_1 \rangle \\ &= \bar{\lambda} \langle w, v_1 \rangle \\ &= 0 \end{aligned}$$
>
> so W is invariant and $A\big|_W$ is a symmetric linear operator on W.
>
> The rest follows by induction.
>
> So we can produce an orthonormal basis of eigenvectors of A and all eigenvalues are real.

Here specifically, the algebraic multiplicity is equal to the geometric multiplicity.

Eigen-Decomposition

Let $v_1, \dots, v_n$ be an orthonormal basis of eigenvectors of A and $\lambda_1, \dots, \lambda_n$ their corresponding eigenvalues.

We can represent $Av_i = \lambda v_i$ with

$$AV = VD$$

where

$$V = \begin{bmatrix} v_1 & \dots & v_n \end{bmatrix}$$

and D is the diagonal matrix with entries $\lambda_1, \dots, \lambda_n$.

So

$$A = VDV^{-1} = VDV^T$$

since

$$V^T V = \begin{bmatrix} v_1^T \\ \dots \\ v_n^T \end{bmatrix} \begin{bmatrix} v_1 & \dots & v_n \end{bmatrix} = I$$

Furthermore

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

which is a sum of outer products.

Matrix Powers

$$A^k = (VDV^T)^k = VD^kV^T$$

which is easily computed.

11.1.2 Orthonormal Basis of Eigenvectors

We can write

$$A^{-1} = \sum_{i=1}^{n} \frac{1}{\lambda_i} v_i v_i^T$$

as long as $\lambda_i \neq 0$.

Even if A is singular, we can produce a "pseudo-inverse" with

$$\sum_{i:\lambda_i \neq 0} \frac{1}{\lambda_i} v_i v_i^T$$

11.1.3 Positive Semidefinite Matrices

Let $A \in \mathbb{R}^{n\times n}$ be symmetric.

Definition 11.1.6 (Positive Semidefinite)
We say A is positive semidefinite if all eigenvalues of A are non-negative.

Remark that this makes sense since A was symmetric and thus has only real eigenvalues.

Proposition 11.1.2
The following are equivalent.

(i) A is positive semidefinite

(ii) $x^T Ax \geq 0$ for all $x \in \mathbb{R}^n$

(iii) $A = UU^T$ for some $U \in \mathbb{R}^{n\times m}$

We write $A \succeq 0$ to indicate that A is PSD.

Proof
$\underline{(1) \implies (3)}$ We know that

$$A = VDV^T$$

let $D^{\frac{1}{2}}$ be the matrix which is the element-wise square root of D. Then

$$A = (VD^{\frac{1}{2}})(D^{\frac{1}{2}}V^T)$$

$\underline{(3) \implies (2)}$ Apply the decomposition to see that

$$\begin{aligned} x^T A x &= x^T U U^T x \\ &= \langle U^T x, U^T x \rangle \\ &= \left\| U^T x \right\|_2^2 \\ &\geq 0 \end{aligned}$$

$\underline{\neg(2) \impliedby \neg(1)}$ If there is some eigenvalue $\lambda < 0$ and v is the corresponding eigenvector,

$$\begin{aligned} 0 &< \lambda \langle v, v \rangle^2 \\ &= \lambda v^T v \\ &= v^T A v \end{aligned}$$

so the quadratic form cannot be non-negative.

11.1.4 Eigenvalue Identities

Proposition 11.1.3
$\mathrm{tr}(A) = \sum_{i=1}^n \lambda_i$ where λ_i are the eigenvalues.

Proof
Remark that

$$\det(xI - A) = (x - \lambda_1) \dots (x - \lambda_n)$$

so looking at the coeffients of x^{n-1}, they must be

$$-\sum_{i=1}^n \lambda_i$$

Looking at this in another direction with the Leibneiz formula, the coefficient of the x^n results from the permutation where $n-1$ diagonal entries are chosen. But then the only permutation to do so must choose the remaining digonal term as well (permutation among row to column).

Thus the term which created x^{n-1} is

$$(x - a_{11}) \dots (x - a_{nn})$$

Using the same idea of inspecting coefficients

$$\sum_i \lambda_i = \sum_i a_{ii}$$

Proposition 11.1.4
$\det A = \prod_{i=1}^{n} \lambda_i$.

11.2 Graph Spectrum

Example 11.2.1
Let $A(G)$ denote the adjacency matrix of G.

If G is complete

$$A(G) = J_n - I_n$$

where J_n is the matrix where every entry is 1.

Now J has rank 1, so its nullspace is of dimension $n-1$ and 0 is an eigenvalue with multiplicity $n-1$.

Notice that $J_n = 1_n 1_n^T$. Thus

$$J_n 1_n = 1_n 1_n^T 1_n = n 1_n$$

and n is an eigenvalue of J with multiplicity one.

The spectrum of J is therefore

$$(n, 0, \ldots, 0)$$

If v is in the nullspace of J

$$(J - I)v = -v$$

Also

$$(J - I)1_n = n1_n - 1_n = (n-1)1_n$$

The spectrum of $A(G)$ is

$$(n-1, -1, \ldots, -1)$$

This is an example with the largest gap between the largest and second largest eigenvalue.

11.3 Bipartite Graphs

Lemma 11.3.1
If G is a bipartite graph and α is an eigenvalue of $A(G)$ with multiplicity k, then $-\alpha$ is also an eigenvalue of $A(G)$ with multiplicity k.

Proof
Let (X, Y) be a bipartition of G. Then we can write

$$A(G) = \begin{bmatrix} 0 & B \\ B^T & 0 \end{bmatrix}$$

Suppose that (x, y) is an eigenvector of A with eigenvalue α.

$$\begin{aligned} \alpha \begin{bmatrix} x \\ y \end{bmatrix} &= A \begin{bmatrix} x \\ y \end{bmatrix} \\ &= \begin{bmatrix} By \\ B^T x \end{bmatrix} \end{aligned}$$

Now consider $(x, -y)$.

$$\begin{aligned} A \begin{bmatrix} x \\ -y \end{bmatrix} &= \begin{bmatrix} -By \\ B^T x \end{bmatrix} \\ &= \alpha \begin{bmatrix} -x \\ y \end{bmatrix} \\ &= -\alpha \begin{bmatrix} x \\ -y \end{bmatrix} \end{aligned}$$

Thus $-\alpha$ is an eigenvalue as well.

Now since the geometric multiplicity is the same as the algebraic multiplicity, a basis of the eigenspace of α gives rise to a basis of the eigenspace of $-\alpha$. Thus equivalence of multiplicity holds.

The converse is also true.

Lemma 11.3.2
$(A^k)_{ij}$ is the number of length k walks from i to j.

Lemma 11.3.3
If $\alpha_1 \geq \alpha_2 \geq \cdots \geq \alpha_n$ are the eigenvalues of $A(G)$ with

$$\alpha_i = \alpha_{n-i+1}$$

for all i, then G is bipartite.

Proof
Notice that if v is an eigenvector of A, then it is also an eigenvector A^k. So the spectrum of A^k where k is odd is

$$\alpha_1^k, \dots, \alpha_n^k$$

But then

$$\operatorname{tr}\left(A^k\right) = 0$$

with each $(A^k)_{ii} \geq 0$ implying that the diagonal of A^k is 0.

So there is no length k walk from i to i for all i and odd k. Specifically, there are no odd cycles.

This shows that G is bipartite.

11.4 Laplacian Matrix

Definition 11.4.1 (Laplacian Matrix)
Let $D(G)$ be the diagonal entries where d_{ii} is the degree of vertex i.
The Laplacian matrix is

$$L(G) := D(G) - A(G)$$

Notice that the Laplacian matrix is also symmetric. Thus the algebraic and geometric multiplicities coincide.

Notice when G is d-regular

$$L(G) = dI - A(G)$$

which has the same spectrum as $A(G)$ (less a constant). In general however, the spectrums could be hard to relate.

A useful way to think about $L(G)$ is

$$L(G) = \sum_{e \in E} L_e$$

where L_e is the Laplacian matrix of the graph e.

$$L_e = e_{ii} + e_{jj} - e_{ij} - e_{ji}$$

Proposition 11.4.1
The quadratic form

$$x^T L x = \sum_{ij \in E} (x_i - x_j)^2$$

Proof

$$\begin{aligned}
x^T L x &= \sum_{e \in E} x^T L_e x \\
&= \sum_{ij \in E} \sum_{k=1}^{n} x_k e_k (e_{ii} + e_{jj} - e_{ij} - e_{ji}) \sum_{\ell=1}^{n} x_\ell e_\ell \\
&= \sum_{ij \in E} \sum_{i=1}^{n} x_k e_k (x_i e_i + x_j e_j - x_j e_i - x_i e_j) \\
&= \sum_{ij \in E} x_i^2 + x_j^2 - x_j x_i - x_i x_j \\
&= \sum_{ij \in E} (x_i - x_j)^2
\end{aligned}$$

11.4.1 Spectrum

Proposition 11.4.2
1_n is an eigenvector of L with eigenvalue 0.

Proof

$$\begin{aligned}
[(D - A)1_n]_i &= \deg(i) - (A1_n)_i \\
&= \deg(i) - \sum_{j : ij \in E} 1 \\
&= \deg(i) - \deg(i) \\
&= 0
\end{aligned}$$

Proposition 11.4.3
The smallest eigenvalue of L is 0.

Proof
We know 0 is an eigenvalue. It suffices to show that $L \succeq 0$.

We know the quadratic form is non-negative. This suffices by our earlier work.

11.4.2 Connectedness

Proposition 11.4.4
A graph is connected if and only if 0 is an eigenvalue of $L(G)$ with multiplicity 1.

Proof
$\underline{(\neg \implies \neg)}$ Suppose G has two components G_1, G_2, with n, m vertices respectively. We have

$$L(G) = \begin{bmatrix} L(G_1) & 0 \\ 0 & L(G_2) \end{bmatrix}$$

so both

$$\begin{bmatrix} 1_n \\ 0_m \end{bmatrix}, \begin{bmatrix} 0_n \\ 1_m \end{bmatrix}$$

are eigenvectors as $1_n, 1_m$ are eigenvectors of $L(G_1), L(G_2)$ respectively.

Moreoever, these two are linearly independent so the geometric multiplicty of 0 is at least 2. This is equivalent to saying the multiplicity is at least 2.

$\underline{(\implies)}$ Suppose now that G is connected. Let x be an eigenvector of eigenvalue 0 of $L(G)$.

Then

$$Lx = 0 \implies x^T Lx = 0$$

But the quadratic form is a sum of non-negative terms $(x_i - x_j)^2$. Thus

$$x_i = x_j$$

for all $ij \in E$. Using a spanning tree and the edges along this tree, we see that $x = c \cdot 1_n$ for some $c \in \mathbb{R}$.

So the geometric multiplicity of 0 is 1. But then the multiplicity of of 0 is also 1.

Second Eigenvalue

Order the eigenvalues

$$\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$$

This shows that G is connected if and only if $\lambda_2 > 0$.

It turns out that G is "close" to being disconnected if and only if λ_2 is small. Moreover G is "close" to having k disconnected components if and only if λ_k is small.

Similarly $\alpha_n \approx -\alpha_1$ if and only if G has a "close to bipartite" component.

11.5 Rayleigh Quotient

As a convention for the eigenvalues of $A(G)$

$$\alpha_1 \geq \alpha_2 \geq \cdots \geq \alpha_n$$

On the other hand,

$$0 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$$

for $L(G)$.

Definition 11.5.1 (Rayleigh Quotient)
Let $x \in \mathbb{R}^n = 0$ and $A \in \mathbb{R}^{n \times n}$ The Rayleigh quotient is

$$R_A(x) := \frac{x^T A x}{x^T x}$$

Lemma 11.5.1
Let A be symmetric, then

$$\alpha_1 = \max_{x \in \mathbb{R}^n} R_A(x)$$

Proof
Let v_1 be an eigenvector corresponding to α_1. Then

$$R_A(v_1) = \alpha_1$$

To see that this is an upperbound, write $x \in \mathbb{R}^n$ as

$$x = \sum_{i=1}^{n} c_i v_i$$

for the orthonormal basis of $\mathbb{R}^n$ ordered the same as the eigenvalues.

> Then
>
> $$\begin{aligned}\langle x, Ax\rangle &= \sum_{i=1}^n c_i^2 \alpha_i \\ &\leq \alpha_1 \sum_{i=1}^n c_i^2 \\ \langle x, x\rangle &= \sum_{i=1}^n c_i^2 \\ R_A(x) &= \frac{\langle x, Ax\rangle}{\langle x, x\rangle} \\ &\leq \alpha_1\end{aligned}$$

In general

Lemma 11.5.2
If A is a real symmetric matrix

$$\alpha_k = \max_{x \in T_k} R_A(x)$$

where

$$T_k = \operatorname{span}\{v_k, \ldots, v_n\}$$

> **Proof**
> For $x \in T_k$ we can write
>
> $$x = \sum_{i=k}^n c_k v_k$$
>
> so
>
> $$\begin{aligned}R_A(x) &= \frac{\sum_{i=k}^n c_i^2 \alpha_i}{\sum_{i=k}^n c_i^2} \\ &\leq \alpha_k\end{aligned}$$
>
> with equality if $x = v_k$.

See the Courant-Fischer theorem without knowing T_k.

11.6 Largest Eigenvalue

Lemma 11.6.1
Let α be the largest eigenvalue of $A(G)$ and v its accompanying eigenvector. Then

$$\alpha \leq \Delta(G)$$

Proof
Let v_j be the maximum entry of v.

$$\begin{aligned}(\alpha v)_j &= (Av)_j \\ &= \sum_{i:ji \in E} v_i \\ &\leq \sum_{i:ji \in E} v_j \\ &\leq \Delta(G) v_j\end{aligned}$$

We have equality if $v_i = v_j$ for all $ji \in E$ and $\deg(j) = \Delta(G)$. Given that G is connected, this means that G is $\Delta(G)$-regular.

Moreover remark that we have shown that the eigenspace of α is 1. Thus the multiplicty of α is 1 and it is the unique largest eigenvalue.

In the case that G is disconnected, this would show that G has a $\Delta(G)$-regular component.

Proposition 11.6.2
The largest α is at least the average degree.

Theorem 11.6.3 (Perron-Frobenius)
Let A be non-negative, irreducible, and aperiodic matrix (not necessarily symmetric).

1. the largest eigenvalue α (in absolute value) has multiplicity 1
2. all entries of the eigenvector corresponding to α are non-zero and have the same sign
3. $|\alpha_i| < \alpha$ for all $1 < i \leq n$

To characterize irreducible and aperiodic, draw an edge ij if $[A]_{ij}$ is non-empty. If that graph is undirected, then it means G is non-bipartite. For directed graphs, it is more difficult to characterize.

Chapter 12

Cheeger's Inequality

12.1 Definitions

12.1.1 Graph Conductance

Recall that G is disconnected if and only if $\lambda_2 = 0$.

We seek to show that $\lambda_2 \approx 0$ if and only if G is "almost" disconnected.

Definition 12.1.1 (Volume)
For $S \subseteq V$

$$\mathrm{vol}(S) := \sum_{v \in S} \deg(v)$$

Definition 12.1.2 (Conductance)
For $\varnothing \neq S \subseteq V$, its conductance is

$$\phi(S) := \frac{|\delta(S)|}{\mathrm{vol}(S)}$$

Clearly

$$\phi(S) \in [0, 1]$$

When G is d-regular

$$\phi(S) = \frac{|\delta(S)|}{d|S|}$$

Definition 12.1.3 (Graph Conductance)
The conductance of a graph is

$$\phi(G) := \min_{S:\mathrm{vol}(S)\leq m} \phi(S)$$

12.1.2 Expander Graphs & Sparse Cuts

Definition 12.1.4 (Expander Graph)
A graph with non-zero conductance is an expander graph.

Definition 12.1.5 (Sparse Cut)
$S \subseteq V$ is a sparse cut if

$$\phi(S)$$

is "small"

12.2 Spectral Partitioning Heuristic

1) Compute an eigenvector $x \in \mathbb{R}^n$ of the second largest eigenvalue of $\mathscr{L}$, the normalized Laplacian
2) Get a linear arrangement of vertices such that $x_i \geq x_{i+1}$
3) Let $S_i = [i]$ if $i \leq \frac{n}{2}$ and $S_i = [n] - [i]$ if $i > \frac{n}{2}$
4) Return the S_i attaining $\min_{i\in[n]} \phi(S_i)$

This is extremely simple and can be implemented in near-linear time.

In practice, it work very well, for example in image segmentation.

12.3 Normalized Matrices

Definition 12.3.1 (Normalized Adjacency Matrix)
Let

$$\mathscr{A} := D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$$

where D is the diagonal degree matrix.

Definition 12.3.2 (Normalized Laplacian Matrix)
Let

$$\mathscr{L} := D^{-\frac{1}{2}} L D^{-\frac{1}{2}}$$

where D is the diagonal degree matrix.

Remark that

$$\mathscr{L} = I - \mathscr{A}$$

When the graph is d-regular

$$\mathscr{A} = \frac{1}{d} A, \mathscr{L} = \frac{1}{d} L$$

Proposition 12.3.1
Let

$$\alpha_1 \geq \cdots \geq \alpha_n$$

be the eigenvalues of $\mathscr{A}$ and

$$\lambda_1 \leq \cdots \leq \lambda_n$$

be the eigenvalues of $\mathscr{L}$.
Then

$$1 = \alpha_1 \geq \cdots \geq \alpha_n \geq -1$$

and

$$0 = \lambda_1 \leq \cdots \leq \lambda_n \leq 2$$

12.4 Cheeger's Inequality

Theorem 12.4.1 (Cheeger)
If λ_2 is the second smallest eigenvalue of $\mathscr{L}$

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

Proof ($\frac{\lambda_2}{2} \leq \phi(G)$)
We only show the case when G is d-regular.

> Recall that 1_n is always an eigenvector of L. By the Rayleigh Quotient
>
> $$\begin{aligned}\lambda_2 &= \min_{x \in T_1} \frac{x^T L x}{d x^T x} \\ &= \min_{x \perp 1_n} \frac{\sum_{ij \in E} (x_i - x_j)^2}{d \sum_{i \in V} x_i^2}\end{aligned}$$
>
> For an arbitrary proper subset $S \subseteq V$ such that $\operatorname{vol}(S) \leq m$, consider $x \in \mathbb{R}^n$ where
>
> $$x_i := \begin{cases} \frac{1}{|S|}, & i \in S \\ \frac{-1}{|V-S|}, & i \notin S \end{cases}$$
>
> so that $x \perp 1_n$.
>
> We have
>
> $$\begin{aligned}\lambda_2 &\leq R_{\mathscr{L}}(x) \\ &= \frac{\sum_{ij \in \delta(S)} \left(\frac{1}{|S|} + \frac{1}{|V-S|} \right)^2}{d \left(|S| \frac{1}{|S|^2} + |V-S| \frac{1}{|V-S|^2} \right)} && \text{cancellation when } ij \notin \delta(S) \\ &= \frac{|\delta(S)| \left(\frac{1}{|S|} + \frac{1}{|V-S|} \right)}{d} \\ &= \frac{|\delta(S)|}{d|S|} \cdot \frac{|V|}{|V-S|} \\ &\leq 2\phi(S) && d|S| = \operatorname{vol}(S) \leq m = \frac{1}{2} dn\end{aligned}$$
>
> This this holds for all S over which we minimize $\phi(S)$ to get conductance, the claim holds.

We should think of λ_2 as a relaxation of the graph conductance problem. Computing the reductance of a graph is NP-hard in general. But relaxing the constraints to "fit" the Rayleigh Quotient, we arrive at λ_2 (within some constant factor)!

One immediate corollary of the easy direction of Cheeger's inequality is a easy way to show that a given graph is an expander graph.

The difficult direction of Cheeger's inequality tells us how to extract a cut from the continuous relaxation of the graph conductance problem. The intuition is that by sorting the vertices according to their entries for a eigenvector of λ_2, we get a vertex ordering where edges are "sufficiently" close.

Proof ($\phi(G) \leq \sqrt{2\lambda_2}$)
Let x be an eigenvector of λ_2. Without loss of generality the number of positive entries is at most the number of negative entries. Let y be the vector where $y_i = \max(x_i, 0)$.

It can be shown that $R(y) \leq R(x)$ where R denotes the Rayleigh Quotient of $\mathscr{L}$ applied at y, x.

We claim that for any such y, there is a subset $S \subseteq \operatorname{supp}(y)$ such that

$$\phi(S) \leq \sqrt{2R(y)}$$

Then

$$\sqrt{2R(y)} \leq \sqrt{2R(x)} = \sqrt{2\lambda_2}$$

since x is an eigenvector corresponding to λ_2. Here

$$\operatorname{supp}(y) := \{i : y_i \neq 0\}$$

Without loss of generality $\|y\|_\infty = 1$ by scaling. For $0 < t \leq 1$, consider the threshold set

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

The goal is to show that there is some t where

$$\phi(S_t) \leq \sqrt{2R(y)}$$

To do so choose $t \in (0, 1)$ uniformly randomly. We claim that

$$E\left[|\delta(S_t)| - d|S_t|\sqrt{2R(y)}\right] \leq 0$$

This would imply that there is t such that

$$\frac{|\delta(S_t)|}{d|S_t|} \leq \sqrt{2R(y)}$$

To compute these values, we apply the linearity of expectation.

$\underline{E[d|S_t|]}$ We have

$$\begin{aligned}
E[d|S_t|] &= dE[|S_t|] \\
&= d\sum_{i\in V} P(i \in S_t) \\
&= d\sum_{i\in V} P(y_i^2 \geq t) \\
&= d\sum_{i\in V} y_i^2
\end{aligned}$$

> $\underline{E[|\delta(S_t)|]}$ Observe that
>
> $$\begin{aligned} E[|\delta(S_t)|] &= \sum_{ij\in E} P(ij \in \delta(S_t)) \\ &= \sum_{ij\in E} P(y_i^2 \geq t > y_j^2) && \text{WLOG } y_i \geq y_j \\ &= \sum_{ij\in E} |y_i^2 - y_j^2| \\ &= \sum_{ij\in E} |y_i - y_j| \cdot |y_i + y_j| \\ &\leq \sqrt{\sum_{ij\in E} (y_i - y_j)^2} \sqrt{\sum_{ij\in E} (y_i + y_j)^2} && \text{Cauchy Schwartz} \\ &\leq \sqrt{R(y) \cdot d \sum_{i\in V} y_i^2} \sqrt{\sum_{ij\in E} (2y_i^2 + 2y_j^2)} \\ &= \sqrt{R(y) \cdot d \sum_{i\in V} y_i^2} \sqrt{\sum_{i\in V} 2d \cdot y_i^2} \\ &= \sqrt{2R(y)} \left(d \sum_{i\in V} y_i^2 \right) \end{aligned}$$
>
> as desired.

Observe that we "randomly rounded" y into an integral solution. This will be further explored in the last part of the course.

12.4.1 Remarks

The proof can be generalized to handle weighted, non-regular graphs.

Cheeger's Inequality provides an

$$O\left(\frac{1}{\sqrt{\phi(G)}}\right)$$

approximation algorithm for $\phi(G)$. This is good if $\phi(G)$ is big but can be very bad if $\phi(G) \approx 0$.

This could be as bad as a $\Theta(n)$-approximation.

To see a tight example of the upper bound given by Cheeger's inequality, consider the cycle

C_n. We can compute

$$\lambda_2 \leq R(x) \in \Theta\left(\frac{1}{n^2}\right)$$

but

$$\phi(C_n) = \Theta\left(\frac{1}{n}\right)$$

Taking 2 copies of C_n and adding edges to give a bijection between the vertices gives a way to "fool" the spectral partitioning algorithm.

Chapter 13

Mixing Time

13.1 Linear Algebraic Formulation of Random Walks

Let $p_t \in \mathbb{R}^n$ be the distribution at time t. For all $v \in V$

$$p_{t+1}(v) = \sum_{u:uv\in E} p_t(u)\frac{1}{\deg u}$$

Let A be the adjacency matrix and D the diagonal degree matrix. Then

$$p_{t+1} = p_t^T D^{-1} A$$

and thus

$$p_t = p_0^T (D^{-1}A)^t.$$

Notice here we used the fact that D is a diagonal matrix or else commutativity might not hold.

It is easier to consider p_t as a row vector thus we can take

$$p_{t+1} = (AD^{-1})p_t$$

Here we used the fact that $(D^{-1}A)^T = A^T D^{-T} = AD^{-1}$.

13.2 Fundamental Theorem of Markov Chains

13.2.1 Stationary Distribution

Let $d \in \mathbb{R}^n$ be the degree vector and $m = |E|$.

Proposition 13.2.1
The distribution $\pi := \frac{d}{2m}$ is a stationary distribution of the random walk on undirected graphs.

Proof
Observe that

$$\begin{aligned}(AD^{-1})\pi &= A\frac{1_n}{2m} \\ &= \frac{d}{2m} \\ &= \pi\end{aligned}$$

It remains to show that it is a probability distribution.

$$\sum_{i=1}^{n} \frac{d_i}{2m} = 1$$

as desired.

Does $p_t \to \pi$ as $\to \infty$? This does not necessarily hold when the Markov chain is reducible and periodic.

13.2.2 Fundamental Theorem of Markov Chains

For undirected graphs, irreducibility just means that the graph is connected. In undirected connected graphs, aperiodic reduces to non-bipartiteness.

Theorem 13.2.2
For any finite, connected, non-bipartite graph

$$p_t \to \pi = \frac{d}{2m}$$

as $t \to \infty$ regardless of p_0.

13.2.3 Spectral Analysis

Let

$$W := AD^{-1}$$

be the random walk matrix and

$$Z := \frac{1}{2}I + \frac{1}{2}W$$

the lazy random walk matrix.

To compute W^t, it is useful to compute the spectrum of W.

Although W is NOT necessarily symmetric, it is similar to a symmetric matrix

$$\begin{aligned} D^{-\frac{1}{2}}AD^{-1}D^{\frac{1}{2}} &= D^{-\frac{1}{2}}AD^{-\frac{1}{2}} \\ &= \mathscr{A} \end{aligned}$$

Proposition 13.2.3
$W, \mathscr{A}$ have the same eigenvalues.

Proof
They have the same characteristic polynomial.

Note that W does not necessarily have an orthonormal basis of eigenvectors. It does however have a basis of eigenvectors.

d-Regular Graphs

In this special case

$$W = \frac{1}{d}A = I - \frac{1}{d}L$$

Theorem 13.2.4
For any finite, connected, non-bipartite d-regular graph

$$p_t \to \pi = \frac{d}{2m}$$

as $t \to \infty$ regardless of p_0.

Proof
Let

$$\alpha_1 \geq \cdots \geq \alpha_n$$

be the eigenvalues of W and

$$v_1, \dots, v_n$$

be an orthonormal basis of eigenvectors.

Write

$$p_0 = \sum c_i v_i.$$

Then

$$W^t p_0 = \sum c_i \alpha_i^t v_1$$

Recall that $-\Delta(G) \leq \alpha_n \leq \alpha_1 \leq \Delta(G)$ for the adjacency matrix. Scaling down shows gives

$$-1 \leq \alpha_n \leq \alpha_1 \leq 1$$

for $\mathscr{A}$.

Moreoever we know that 1_n is an eigenvector of $\mathscr{A}$, thus

$$\alpha_1 = 1$$

Recall also that $0 = \lambda_1 < \lambda_2$ if and only if G is connected for the Laplacian matrix. Since $A = 1 - L$,

$$\alpha_1 = 1 > \alpha_2$$

given that G is connected.

We also know that $\alpha_1 = -\alpha_n$ if and only if the graph is bipartite. Thus

$$\alpha_n < -1$$

in our case.

This means that

$$W^t p_0 \to c_1 v_1.$$

Now 1_n is an eigenvector of L with eigenvalue 0. Thus $\frac{1_n}{\sqrt{n}}$ is an eigenvector of $\mathscr{L}$ with eigenvalue 1 when G is d-regular. But then it is also an eigenvalue of $\mathscr{A}$ with eigenvalue 1.

So $v_1 = \frac{1_n}{\sqrt{n}}$.

We have

$$\begin{aligned} c_1 &= \left\langle p_0, \frac{1_n}{\sqrt{n}} \right\rangle \\ &= \frac{1}{\sqrt{n}} \langle p_0, 1_n \rangle \\ &= \frac{1}{\sqrt{n}} \\ c_1 v_1 &= \frac{1}{\sqrt{n}} \cdot \frac{1_n}{\sqrt{n}} \\ &=: \pi \end{aligned}$$

as required.

Suppose that

$$|\alpha_i| \leq 1 - \epsilon$$

for $2 \leq i \leq n$.

Then

$$\begin{aligned}
\alpha_i^t &\leq (1-\epsilon)^t \\
&\leq e^{-\epsilon t} \\
&\leq \frac{1}{n^p} && t := p \ln\left(\frac{n}{\epsilon}\right)
\end{aligned}$$

so the convergence is very fast.

13.2.4 Lazy Random Walks

While finiteness and connectedness seem natural, non-bipartiteness does not.

We can remove the non-bipartiteness assumption by doing lazy random walks.

Definition 13.2.1 (Lazy Random Walk)
In each iteration, we stay at the same vertex with probability $\frac{1}{2}$ and move to a uniform random neighbour with probability $\frac{1}{2}$.

The transition matrix is modified to become

$$p_t = \left(\frac{1}{2}I + \frac{1}{2}AD^{-1}\right)^t p_0.$$

Theorem 13.2.5
For any finite connected graph with lazy random walks,

$$p_t \to \pi = \frac{d}{2m}$$

as $t \to \infty$ regardless of p_0.

Intuitively adding a self-loop with large probability makes the graph "VERY" aperiodic.

Proof
Let the spectrum of W be

$$1 \geq \alpha_1 \geq \cdots \geq \alpha_n \geq -1.$$

> The spectrum of Z is just
>
> $$1 = \frac{1}{2}(1+\alpha_1) \underbrace{>}_{\text{connectedness}} \cdots \ge \frac{1}{2}(1+\alpha_n) \ge 0.$$

Notice that $\frac{1}{2}$ is not very crucial. It is chosen for simplicity sake.

13.3 Mixing Time

We would like to understand how fast p_t converges to π.

A standard measure is the scaled 1-norm

$$\frac{1}{2}\|p_t - \pi\|_1 = \frac{1}{2}\sum_{i=1}^{n}|p_t(i) - \pi(i)|$$

13.3.1 Mixing Time by Spectral Gap

Definition 13.3.1 (ϵ-Mixing Time)
The ϵ-mixing time of the random walk is defined as the smallest t such that

$$\|p_t - \pi\|_1 \le \epsilon$$

for all p_0.

Definition 13.3.2 (Spectral Gap)
The spectral gap is

$$\lambda := \min(1-\alpha_2, 1-|\alpha_n|)$$

Observe that

$$|\alpha_i| \le 1 - \lambda$$

for $2 \le i \le n$.

We will bound the mixing time using the spectral gap.

Theorem 13.3.1
The ϵ-mixing time is upper bounded by

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

where λ is the spectral gap.

Proof (*d*-Regular Graphs)
Write

$$p_t = W^t p_0 = \sum_{i=1}^n c_i \alpha_i^t v_i$$

Then since G is d-regular

$$\begin{aligned}
\|p_t - \pi\|_2^2 &= \Big\|\sum_{i=2}^n c_i \alpha_i^t v_i\Big\|_2^2 \\
&= \left\langle \sum_{i=2}^n c_i \alpha_i^t v_i, \sum_{i=2}^n c_i \alpha_i^t v_i \right\rangle \\
&= \sum_{i=2}^n c_i^2 \alpha_i^{2t} \|v_i\|_2^2 \\
&\leq (1-\lambda)^{2t} \sum_{i=2}^n c_i^2 \\
&\leq (1-\lambda)^{2t} \sum_{i=1}^n c_i^2 \\
&\leq (1-\lambda)^{2t} \|p_0\|_2^2 \\
&\leq (1-\lambda)^{2t} \|p_0\|_1^2 \\
&\leq (1-\lambda)^{2t}.
\end{aligned}$$

By the Cauchy-Schwarz inequality and the fact that G is d-regular

$$\begin{aligned}
\|p_t - \pi\|_1 &\leq \sqrt{n}\|p_t - \pi\|_2 \\
&\leq \sqrt{n}(1-\lambda)^t \\
&\leq \sqrt{n}e^{-\lambda t} \\
&\leq \epsilon & t \geq \frac{1}{\lambda}\log\left(\frac{\sqrt{n}}{\epsilon}\right)
\end{aligned}$$

The statement holds in general but it is n instead of $\sqrt{n}$.

13.3.2 Mixing Time for Lazy Random Walks

In lazy random walks the spectral gap is simply

$$\frac{\lambda_2}{2}$$

where λ_2 is the second eigenvalue of $\mathscr{L}$.

From Cheeger's inequality, we know that

$$\lambda_2 \geq \frac{\phi(G)^2}{2}$$

Theorem 13.3.2
The ϵ-mixing time of lazy random walks is upper bounded by

$$\frac{2}{\phi(G)^2} \log\left(\frac{n}{\epsilon}\right).$$

This implies that lazy random walks mix fast in an expander graph.

13.4 Random Sampling

13.4.1 Card Shuffling

How many times do we need to shuffle a deck of cards so that the distribution almost uniform.

Say we have a deck of 52 cards. How can be we obtain a random permutation using simple operations?

fix some basic operations. Does applying basic operations converge the distribution of all permutations? If so, how many steps are required to get an almost uniform distribution?

We can understand these questions about random walks on "state" graphs.

A famous result is that the 7 shiffle shuffle results in a convergent sequence.

13.4.2 Random Perfect Matching in Bipartite Graphs

Sampling over all perfect matchings.

This is related to the permanant of a matrix.

13.4.3 Random Spanning Tree

There may be exponentially many spanning trees. We want to sample an arbitrary one.

We can get from spanning tree to spanning tree by adding and deleting an edge. There is a recent result which shows that we can get a uniform spanning tree in time

$$\tilde{O}(m)$$

13.4.4 Approximate Counting

There is a reduction from sampling to this application.

This is a class of problems for which we know polynomial time probabilistic algorithms but NOT deterministic algorithms.

Chapter 14

Electrical Networks

14.1 Electrical Flow

Definition 14.1.1 (Electrical Network)
An edge-weighted undirected graph where each edge is seen as a resistor of resistance r_e.

Definition 14.1.2 (Kirchoff's Law)
The sum of incoming currents is equal to sum of outgoing currents.

This is just the standard flow conservation law.

Definition 14.1.3 (Ohm's Law)
There exist a voltage vector $\phi : V \to \mathbb{R}$ such that

$$\phi(u) - \phi(v) = f_{uv} r_{uv}$$

for all $e = uv \in E$.

It is important to note that the voltage is an undirected quantity while the flow IS a directed quantity. Namely

$$f_{vu} = f_{uv}.$$

Definition 14.1.4 (Electrical Flow)
An electrical flow is a flow which obeys Kirchoff and Ohm's Laws.

We need to indicate source and sink nodes to and from which we inject and extract flow.

14.1.1 Linear Equations

Write

$$w_e := \frac{1}{r_e}$$

to be the conductance of the edge e.

Associate with each vertex $v \in V$ a demand b_v. This is positive if we are injecting flow and negative if we are extracting flow.

To satisfy Ohm's Law

$$f_{uv} = w_{uv}(\phi(u) - \phi(v))$$

for all $uv \in E$.

Definition 14.1.5 (Weighted Degree)
The weighted degree of $v \in V$ is

$$\deg_w(v) := \sum_{u:uv\in E} w_{vu}.$$

In order to obey Kirchoff's Law

$$\begin{aligned} b_v &= \sum_{u:uv\in E} f_{vu} \\ &= \sum_{u:uv\in E} w_{vu}(\phi(v) - \phi(u)) \\ &= \phi(v)\deg_w(v) - \sum_{u:uv\in E} \phi(u) w_{vu} \end{aligned}$$

for all $v \in V$.

14.1.2 Matrix Formulation

Suppose that $w_{uv} = 1$ for each $uv \in E$. Then the restriction becomes

$$\vec{b} = L\vec{\phi}$$

where L is the Laplacian matrix of G.

Definition 14.1.6 (Weighted Laplacian Matrix)
Define the weighted Laplacian matrix as

$$L_{uv} = \begin{cases} -w_{uv}, u \neq v, uv \in E \\ 0, & u \neq v, uv \notin E \\ \deg_w(v), u = v \end{cases}.$$

Then we again have the equation

$$\vec{b} = L\vec{\phi}.$$

14.1.3 Computing Voltages & Flows

Once the voltage is set, the flow is also uniquely determined.

Let B be the $n \times m$ matrix

$$B = \left[b_{e_1}, b_{e_2}, \dots, b_{e_m}\right]$$

where b_{uv} is the vector which is $+1$ in the u-th entry and -1 in the v-th entry.

Remark that flow can be computed by

$$f = B^T \phi.$$

Moreover

$$L = \sum_{e \in E} w_e b_e b_e^T = BWB^T$$

where W is the $m \times m$ diagonal matrix of conductance.

Thus

$$\vec{b} = L\vec{\phi} = BWB^T\vec{\phi} = BWf.$$

Now if $W = I$. Then

$$\vec{b} = B\vec{f}$$

which is just the flow conservation constraints in matrix form.

14.1.4 Solution Space

L is not of full rank since 0 is an eigenvalue. But if G is without loss of generality connected, the the nullspace of L is spanned by 1_n.

Proposition 14.1.1
If $Lx = b$, then
$$b \perp 1 - n.$$

Proof
Write
$$x = \sum_{i=1}^{n} c_i v_i$$
where $\{v_i\}$ is the basis of eigenvectors of L.

Then
$$Lx = \sum_{i=1}^{n} c_i \lambda_i v_i = \sum_{i=2}^{n} c_i \lambda_i v_i \perp 1_n.$$

This makes sense for electrical flows, since the demand should sum to 0.

Proposition 14.1.2
If $b \perp 1_n$, then there is x such that
$$Lx = b.$$

Proof
Write
$$b = \sum_{i=2}^{n} a_i v_i.$$

Then
$$L\left(\sum_{i=2}^{n} \frac{a_i}{\lambda_i} v_i\right) = x.$$

14.1.5 Pseudo-Inverse

Definition 14.1.7 (Pseudo-Inverse)
The Pseudo-Inverse of the Laplacian is
$$L^\dagger := \sum_{i=2}^{n} \frac{1}{\lambda_i} v_i v_i^T.$$

It follows that the solution set for $Lx = b$ is

$$\{L^\dagger b + c1_n : c \in \mathbb{R}\}$$

which is a “shift” of the solutions.

In particular, the flow $\vec{f}$ is unique!

Moreover, any Laplacian system can be thought of as an electrical flow problem!

14.2 Effective Resistance

Definition 14.2.1 (Effective Resistance)
Set $b_s = 1, b_t = -1$ and b is zero otherwise. Solve $L\phi = b$.
The effective resistance is

$$R_{\text{eff}}(s,t) := \phi(s) - \phi(t).$$

Think of it as the resistance of the entire graph as a resistor.

Proposition 14.2.1
The effective resistance is

$$R_{\text{eff}}(s,t) = b_{st}^T L^\dagger b_{st}$$

where

$$b_{st}(s) = 1, b_{st}(t) = -1$$

and zero otherwise.

14.2.1 Energy

Definition 14.2.2 (Energy)
The energy of an electrical flow is

$$\varepsilon(\vec{f}) := \sum_{e \in E} f_e^2 r_e.$$

Proposition 14.2.2
We claim that

$$\varepsilon(\vec{f}) = R_{\text{eff}}(s,t)$$

where f is a 1-unit electrical flow from s to t.

Proof
We have

$$\begin{aligned}
\varepsilon(\vec{f}) &= \sum_e f_e^2 r_e \\
&= \sum_{uv \in E} w_{uv}(\phi(u) - \phi(v))^2 \\
&= \phi^T L \phi \\
&= b_{st}^T L^\dagger L L^\dagger b_{st} && \phi = L^\dagger b_{st} \\
&= b_{st}^T L^\dagger b_{st} \\
&=: R_{\text{eff}}(s,t).
\end{aligned}$$

14.2.2 Thompson's Principle

Theorem 14.2.3 (Thompson's Principle)
Let $\vec{g}$ be a 1-unit st-flow (not necessarily electrical).
Then

$$R_{\text{eff}}(s,t) \leq \varepsilon(\vec{g}).$$

Thus the one unit st-electrical flow is the flow that minimizes the energy among all 1-unit st-flows.

Proof
Recall that the minimization problem

$$\min g(x), Ax = b$$

has an optimality condition which says any optimal solution x^* has a corresponding y such that

$$\boldsymbol{\nabla} g(x^*) = A^T y.$$

Then the problem

$$\sum_e f_e^2, Bf = b_{st}$$

has the optimal condition that for all optimal flows f, there is some $\phi \in \mathbb{R}^n$ such that

$$2\vec{f} = B^T \phi.$$

Thus f is defined by the voltage vector $\frac{\phi}{2}$.

14.2.3 Rayleigh's Monotonicity Principle

Theorem 14.2.4 (Rayleigh's Monotonicity Principle)
If $\vec{r'} \geq \vec{r}$, then

$$R_{\text{eff},\vec{r'}}(s,t) \geq R_{\text{eff},\vec{r}}(s,t).$$

Intuitively, this just says increasing the resistance of a single edge does not increase the effective resistance of the graph as a whole.

Proof
This is a corollary of Thompson's Principle.

$$\begin{aligned} R_{\text{eff},r'}(s,t) &= \varepsilon_{r'}(f') \\ &\geq \varepsilon_r(f') && \sum f_e^2 r_e \\ &\geq \varepsilon_r(f) && \text{Thompson's Principle} \\ &= R_{\text{eff},r}(s,t). \end{aligned}$$

14.2.4 Short Disjoint Paths

Lemma 14.2.5
If there are k vertex-disjoint st-paths, each of length at most ℓ, then $R_{\text{eff}}(s,t) \leq \frac{\ell}{k}$.

Proof
Let G be the original paths. Let G_1 be the union of disjoint s,t-paths. Then let G_2 be the graph in which we artificially subdivide edges until each path is exactly of length ℓ.

We can think of edge deletion as increasing the resistance to ∞. In addition, edge subdivision is also increases the resistance. Thus

$$R_{\text{eff},G}(s,t) \leq R_{\text{eff},G_1}(s,t) \leq R_{\text{eff},G_2}(s,t) \leq \frac{\ell}{k}.$$

14.2.5 Effective Resistance Metric

In some ways, effective resistance may be a better metric to measure how close two nodes are.

It is known that effective resistance satisfies the triangle inequality.

Lemma 14.2.6
$R_{\text{eff}}(a,b) + R_{\text{eff}}(b,c) \geq R_{\text{eff}}(a,c)$ for all $a, b, c \in V$.

14.3 Random Walks

We study some interesting quantities about random walks in undirected graphs.

Definition 14.3.1 (Hitting Time)
$h_{u,v} = E[H_{u,v}]$ where

$$H_{u,v} := \min\{t \geq 0 : X_1 = u, X_t = v\}.$$

Definition 14.3.2 (Commute Time)
$c_{u,v} = h_{u,v} + h_{v,u}$

Definition 14.3.3 (Cover Time)
$\text{cover}_g := \max_v \text{cover}_v$ where cover_v is the expected time to visit every vertex at least once if the random walk starts at v.

14.3.1 Commute Time

Theorem 14.3.1
For any two vertices s, t

$$c_{s,t} = 2mR_{\text{eff}}(s,t)$$

where $m = |E(G)|$.

Proof
The goal is to show that the two quantities satisfy the same set of equations.

Observe that

$$h_{v,t} = \frac{1}{d(v)} \sum_{w:vw \in E} (1 + h_{wt})$$

for any $v \in V - t$ and $h_{tt} = 0$.

This is equivalent to

$$\begin{aligned}
d(v)h_{vt} &= d(v) + \sum_{w:vw\in E} h_{wt} \\
d(v) &= d(v)h_{vt} - \sum_{w:vw\in E} h_{wt} \\
&= \sum_{w:vw\in E} h_{vt} - h_{wt} \\
&= \sum_{w:vw\in E} h(v) - h(w)
\end{aligned}$$

for $v \in V - t$.

This is ALMOST a Laplacian system of linear equations. Indeed, consider the electrical flow problem defined by demand b_t where we inject $d(v)$ units of current to each $v \in V - t$ and remove $2m - d(t)$ units of electrical flow from t.

Let ϕ_{vt} be the voltage at v in this electrical flow with $\phi_{tt} = 0$. There is then a unique solution.

We must have

$$\phi_{vt} = h_{vt}$$

and thus

$$L\vec{h}_t = \vec{b}_t.$$

In similar fashion, the electrical flow problem defined by demand b_s where we inject $d(v)$ units of current for each $v \in V - s$ and remove $2m - d(s)$ units of electrical flow from s corresponds to the constraints for $h_{v,s}$.

So

$$L\vec{h}_s = \vec{b}_s.$$

It follows that

$$\begin{aligned}
L(\vec{h}_t - \vec{h}_s) &= \vec{b}_t - \vec{b}_s \\
&= 2m(\chi_s - \chi_t) \\
\frac{1}{2m}(\vec{h}_t - \vec{h}_s) &= L^\dagger(\chi_S - \chi_t).
\end{aligned}$$

Thus

$$\frac{1}{2m}(\vec{h}_t - \vec{h}_s)$$

is the voltage vector when one unit of electrical flow is sent from $s \to t$.

Finally, we see that

$$\begin{aligned}R_{\text{eff}}(s,t) &= (\chi_s - \chi_t)^T \frac{1}{2m}(\vec{h}_t - \vec{h}_s) \\ &= \frac{1}{2m}(h_{ts} - h_{st}) \\ &= \frac{c_{st}}{2m}.\end{aligned}$$

Corollary 14.3.1.1
$c_{u,v} \leq 2m$ for every edge $uv \in E$.

Proof
We have

$$R_{\text{eff}}(s,t) \leq 1$$

for all $uv \in E$ as there is 1 vertex-disjoint st-path of length 1.

14.3.2 Cover Time

Theorem 14.3.2
The cover time of a connected graph is at most

$$2m(n-1).$$

Proof
Take a spanning tree T of G. Starting at any vertex v, consider the tour C where we walk along the outer face of the tree.

Then

$$\begin{aligned}\text{cover}_v &\leq E[C] \\ &= \sum_{uv \in T} h_{uv} + h_{vu} \\ &= \sum_{uv \in T} c_{uv} \\ &\leq \sum_{uv \in T} 2m \\ &= 2m(n-1).\end{aligned}$$

Resistance Diameter

Definition 14.3.4 (Resistance Diameter)
This is the maximum effective resistance between two vertices.

$$R(G) := \max_{u,v\in V} R_{\text{eff}}(u,v).$$

Theorem 14.3.3
We have

$$mR(G) \leq \text{cover}(G) \leq 2e^3 mR(G)\ln n + n.$$

Proof
Let u, v be the nodes attaining the resistance diameter R. From the previous theorem

$$2mR = c_{u,v} = h_{uv} + h_{vu}.$$

Without loss of generality, $h_{uv} \geq mR$. It follows that

$$\text{cover}_u \geq h_{uv} \geq mR.$$

For ANY $u, v \in V$

$$h_{u,v} \leq c_{uv} \leq 2mR.$$

Thus by Markov's Inequality

$$P(v \text{ not covered in } 2e^3 mR \text{ steps}) \leq \frac{h_{u,v}}{2e^3 mR} \leq \frac{1}{e^3}.$$

But observe that $h_{s,v} \leq 2mR$ so after each $2e^3mR$ steps, we have an independent chance to hit v within another $2e^3mR$ steps. It follows that

$$P(v \text{ not covered in } 2e^3 mR\ln n \text{ steps}) \leq (e^{-3})^{\ln n} = \frac{1}{n^3}.$$

Then by a union bound, the probability that no vertex is covered in $2e^3mR\ln n$ steps is at most

$$\frac{1}{n^2}.$$

Applying the simpler bound from before

$$\text{cover}(G) \leq \left(1 - \frac{1}{n^2}\right) 2e^3 mR\ln n + \frac{1}{n^2}\cdot n^3$$

as desired.

Graph Connectivity

Proposition 14.3.4
There is an $O(n^3)$ time algorithm which solves the *st*-connectivity problem using only $O(\log n)$ space.

Proof
Just walk randomly for $O(n^3)$ steps starting from s.

Chapter 15

Spectral Sparsification

15.1 Spectral Sparsification

15.1.1 Cut Sparsifiers

Definition 15.1.1 (ϵ-Cut Approximation)
A graph H is an ϵ-cut approximation of G if for all $S \subseteq V$

$$(1-\epsilon)w\delta_G(S) \leq w\delta_H(S) \leq (1+\epsilon)w\delta_G(S).$$

Theorem 15.1.1
Any graph G has an ϵ-cut approximator H with

$$O\left(\frac{n \log n}{\epsilon^2}\right)$$

edges.

A small remark that a near0linear time algorithm is one which runs in $\tilde{O}(n)$ time and an almost-linear time algorithm is one which runs in $O(n^{1+O(1)})$ time.

15.1.2 Spectral Sparsifiers

Recall that $A \succeq B$ means that

$$A - B \succeq 0$$

is a positive semi-definite matrix.

Definition 15.1.2 (ϵ-Spectral Approximator)
A graph H is an ϵ-spectral approximator of G if

$$(1-\epsilon)L_G \preceq L_H \preceq (1+\epsilon)L_G.$$

Here L_G can be the Laplacian matrix of G or the weighted Laplacian matrix if G has weighted edges.

An equivalent definition of $A \preceq B$ is if

$$x^T A x \leq x^T B x$$

for all $x \in \mathbb{R}^{n\times n}$.

Proposition 15.1.2
An ϵ-spectral approximator is an ϵ-cut approximator.

Proof
Let $\chi_S \in \mathbb{R}^n$ be the characteristic vector of some $S \subseteq V$.

$$\begin{aligned}(1-\epsilon)w\delta_G(S) &= (1-\epsilon)\sum_{uv\in E}(\chi_S(u)-\chi_S(v))^2 \\ &= (1-\epsilon)\chi_S^T L_G \chi_S \\ &\leq \chi_S^T L_H \chi_S \\ &\leq (1+\epsilon)\chi_S^T L_G \chi_S \\ &= (1+\epsilon)w\delta_G(S)\end{aligned}$$

as desired.

Theorem 15.1.3
Any graph G has an ϵ-spectral approximator H with

$$O\left(\frac{n\log n}{\epsilon^2}\right)$$

edges.

Linear Algebraic Reduction

There is a reduction from the spectral sparsification problem to a purely linear algebraic problem.

Theorem 15.1.4
Suppose $v_1, \dots, v_m \in \mathbb{R}^n$ are given such that

$$\sum_{i=1}^{m} v_i v_i^T = I_n.$$

There are scalars $s_1, \dots, s_m$ with at most

$$O\left(\frac{n \log n}{\epsilon^2}\right)$$

non-zero entries in total such that

$$(1-\epsilon) \cdot I_n \preceq \sum_{i=1}^{m} s_i v_i v_i^T \preceq (1+\epsilon) \cdot I_n.$$

Isotropy Condition

Definition 15.1.3 (Isotropy Condition)
The condition

$$\sum_{i=1}^{m} v_i v_i^T = I_n$$

is called the isotropy condition.

We can think of this as an overcomplete basis.

For all unit vectors y.

$$\begin{aligned}\sum_{i=1}^{m} \langle y, v_i \rangle \langle v_i, y \rangle &= y^T \sum_{i=1}^{m} v_i v_i^T y \\ &= y^T I y \\ &= 1\end{aligned}$$

Sampling Algorithm

1) Initially set $F := \varnothing, \vec{s} := 0_m, C := \frac{6 \log n}{\epsilon^2}$
2) For $1 \leq t \leq C$, for each $1 \leq i \leq m$ with probability $p_i = \|v_i\|_2^2$, add $F = F \cup \{i\}$ and $s_i = s_i + \frac{1}{Cp_i}$
3) Return $\sum_{i \in F} s_i v_i v_i^T$

Number of Vectors

Lemma 15.1.5
With probability at least 0.9

$$|F| = O\left(\frac{n \log n}{\epsilon^2}\right).$$

Proof
By the linearity of the trace

$$\begin{aligned}
\sum_{i=1}^m \|v_i\|_2^2 &= \sum_{i=1}^m \operatorname{tr}\left(v_i v_i^T\right) \\
&= \sum_{i=1}^m \operatorname{tr}\left(v_i v_i^T\right) \\
&= \operatorname{tr} I_n \\
&= n
\end{aligned}$$

We have

$$\begin{aligned}
E[|F|] &= \sum_{i=1}^m P(v_i \text{ is chosen}) \\
&\leq \sum_{i=1}^m C p_i \\
&= C \sum_{i=1}^m \|v_i\|_2^2 \\
&= \frac{6 \log n}{\epsilon^2} \sum_{i=1}^m \|v_i\|_2^2 \\
&= \frac{6n \log n}{\epsilon^2}.
\end{aligned}$$

Applying Markov's inequality concludes the proof.

Matrix Chernoff Bound

Theorem 15.1.6
Let $X_1, \dots, X_m$ be independent $n \times n$ real symmetric matrices such that

$$0 \preceq X_i \preceq R \cdot I$$

for some $r \in \mathbb{R}$.
Let

$$\mu_{\min} I \preceq \sum_{i=1}^{m} E[X_i] \preceq \mu_{\max} I.$$

For any $0 < \epsilon \leq 1$

$$P\left(\lambda_{\max}\left(\sum_{i=1}^{m} X_i\right) \geq (1+\epsilon)\mu_{\max}\right) \leq ne^{-\frac{\epsilon^2 \mu_{\max}}{3R}}$$
$$P\left(\lambda_{\min}\left(\sum_{i=1}^{m} X_i\right) \leq (1-\epsilon)\mu_{\min}\right) \leq ne^{-\frac{\epsilon^2 \mu_{\min}}{2R}}.$$

Concentration

The random variables are

$$X_{i,t} = \frac{v_i v_i^T}{Cp_i}$$

with probability p_i and 0 otherwise.

Now

$$\begin{aligned} E[\text{Output}] &= \sum_{t=1}^{C} \sum_{i=1}^{m} E[X_{i,t}] \\ &= \sum_{t=1}^{C} \sum_{i=1}^{m} \frac{v_i v_i^T}{C} \\ &= \sum_{i=1}^{m} v_i v_i^T \\ &= I \end{aligned}$$

hence $\mu_{\max} = \mu_{\min} = 1$.

Observe that

$$\begin{aligned} X_{i,t} &= \frac{v_i v_i^t}{C p_i} \\ &= \frac{1}{C}\left(\frac{v_i}{\|v_i\|}\right)\left(\frac{v_i}{\|v_i\|}\right)^T. \end{aligned}$$

So $X_{i,t}$ is a rank 1 matrix whose only eigenvector is $\frac{v_i}{\|v_i\|}$ with eigenvalue $\frac{1}{C}$.

We can thus take

$$R = \frac{1}{C}.$$

An application of the matrix Chernoff bound gives

$$\begin{aligned} P\left(\lambda_{\max}\left(\sum_{i=1}^m s_i v_i v_i^T\right) \geq 1+\epsilon\right) &\leq n e^{-\frac{\epsilon^2 C}{3}} \\ &= n e^{-2\ln n} \\ &= \frac{1}{n}. \end{aligned}$$

15.1.3 Linear-Sized Spectral Sparsifiers

Theorem 15.1.7
Any graph G has an ϵ-spectral approximator H with

$$O\left(\frac{n}{\epsilon^2}\right)$$

edges.

The proof is purely linear algebraic and gives a deterministic algorithm to construct a sparsifier.

There are near-linear time algorithms to find a linear-sized spectral sparsifier now.

15.2 Effective Resistance

Recall that the Laplacian is $L_G = \sum_e b_e b_e^T$.

The reduction from spectral sparsification to linear algebra is

$$v_e = L_G^{\frac{\dagger}{2}} b_e.$$

Thus the sampling probability is

$$\begin{aligned} p_e &= \|v_e\|_2^2 \\ &= \left\| L_G^{\frac{\dagger}{2}} \right\|_2^2 \\ &= b_e^T L_G^\dagger b_e \\ &= R_{\text{eff}}(e) \end{aligned}$$

It is possible to compute good approximations of the sampling probabilities in near-linear time. The idea is to do dimension reduction.

Part III

Linear Programming

Chapter 16

Linear Programming

16.1 Review

16.1.1 Linear Program

Any linear program can be expressed in inequality form

$$\begin{aligned} \max\langle c, x\rangle \\ Ax \leq b. \end{aligned}$$

We generally expect $m \geq n$ in this case.

An equivalent equality form is

$$\begin{aligned} \max\langle c, x\rangle \\ Ax = b \\ x \geq 0. \end{aligned}$$

We generally expect $n \leq m$ in this case.

16.1.2 Integer Program

Many combinatorial optimization problems can be formulated as an integer linear program.

Maximum weight bipartite matching can be written as

$$\begin{aligned} \max \sum_{e \in E} c_e x_e & \\ \sum_{e \in \delta(v)} x_e \leq 1 & \qquad v \in V \\ x_e \in \{0, 1\} & \qquad e \in E. \end{aligned}$$

Maximum independent set can be expressed as

$$\begin{aligned} \max \sum_{v \in V} c_v x_v & \\ x_u + x_v \leq 1 & \qquad uv \in E \\ x_v \in \{0, 1\}. & \qquad v \in V \end{aligned}$$

16.1.3 LP Relaxation

The idea is to relax the constraints $x \in \{0, 1\}^n$ to linear constraints $0 \leq x \leq 1$.

However, the result can be very bad.

For some problems, such as the formulation for bipartite matching, can be solved EXACTLY with LPs!

This is because all extreme points of the feasible region are integral and we can find basic solutions corresponding to these points.

16.1.4 Corner Points

Equality Form

There are 3 equivalent characterizations of "corner points" for LPs in equality form.

Let $P := \{Ax = b, x \geq 0\}$.

Vertex Solution $x \in P$ is a vertex solution if no $y \neq 0$ is such that $x + y, x - y \in P$

Extreme Point Solution $x \in P$ is an extreme point solution if there is some c such that x is the UNIQUE optimal solution of $\max\langle c, x \rangle$ for $x \in P$

Basic Solution let $\text{supp}(x) := \{i : x_i \neq 0\}$ $x \in P$ is a basic solution if the columns corresponding to $\text{supp}(x)$ are linearly independent.

Inequality Form

Let $P := \{Ax \leq b\}$.

We say the i-th constraint is tight is

$$\langle A_i, x\rangle = b_i$$

where A_i is the i-th row.

Given $x \in P$, let $A^=$ be the submatrix of A formed by tight constraints.

Vertex Solution $x \in P$ is a vertex solution if no $y \neq 0$ is such that $x+y, x-y \in P$

Extreme Point Solution $x \in P$ is an extreme point solution if there is some c such that x is the UNIQUE optimal solution of $\max\langle c, x\rangle$ for $x \in P$

Basic Solution $x \in P$ is a basic solution if $A^=$ is of full rank ($\operatorname{rank} A^= = n$).

Optimal Corner Point

Proposition 16.1.1
For bounded P and $c \in \mathbb{R}^n$ there is a basic feasible solution such that

$$\langle c, x\rangle \geq \langle c, y\rangle$$

for all $y \in P$.

Proof
Let $x \in P$ be an optimal solution satisfying the the maximal number of constraints. Suppose that

$$\operatorname{rank} A^= < n.$$

There is some $0 \neq y \in \mathbb{R}^n$ such that

$$A^= y = 0.$$

There is some $\epsilon > 0$ such that

$$A(x+\epsilon y), A(x-\epsilon y) \leq b$$

We must have both

$$\langle c, x\rangle \geq \langle x+\epsilon y\rangle, \langle x-\epsilon y\rangle$$

by optimality. In particular $c^T y = 0$.

> But then
> $$\{x + \lambda y : \lambda \in \mathbb{R}\} \cap P$$
> is a set of optimal solutions. Since P is bounded, there is a maximal/minimal λ such that $x + \lambda y \in P$.
>
> In other words, $x + \lambda y$ is optimal while satisfying at least one more constraints than x. This is the desired contradiction.

16.2 Perfect Bipartite Matching

Theorem 16.2.1
The following LP is integral for perfect matching in a bipartite graph

$$\begin{aligned} &\max \sum_{e \in E} c_e x_e \\ &Mx = 1 && v \in V \\ &x \geq 0 \\ &x \leq 1 \end{aligned}$$

where M is the unsigned incidence matrix of G.

> **Proof**
> M is totally unimodular.

16.3 Solving Linear Programs

16.3.1 Simplex Method

Start from an arbitrary basic solution and move from vertex to vertex until we attain optimality.

16.3.2 Ellipsoid Method

The idea is to reduce optimization to a decision problem, ie checking if P is empty.

Begin with a sufficiently large ellipsoid E_0 containing P.

Given E_i, check if the center of E_i resides in P. If so, we are done. If not, require a separating hyperplane H as certificate.

Find an ellipsoid E_{i+1} containing $E_i \cap H$. In particular, $P \subseteq E_{i+1}$.

The key point is to show that

$$\frac{\operatorname{vol} E_{i+1}}{\operatorname{vol} E_i} \leq e^{-\frac{1}{2(n+1)}}$$

so after $O(n)$ steps, the volume has decreased by a constant fraction.

this belongs to the class of algorithms called cutting plane methods.

Optimization via Separation

An important feature of the ellipsoid method is that it only requires a "separation" oracle for it to work.

In particular, we do not need to write down the LP explicitly.

This means we can solve exponentially sized LPs (with respect to the input size) in polynomial time with the ellipsoid method!

16.3.3 Interior Point Method

First use "barrier functions" to ensure that an optimal solution stays in the polytope. This reduces to an unconstrained optimization problem.

Initially, we start from the "center" of the polytope. In each step, we decrease the "force" of the barrier and update the current point using Newton's method.

Chapter 17

Matching Polytopes

17.1 Bipartite Matching

Consider the weighted bipartite matching formulation

$$\begin{aligned} &\max\langle c, x\rangle && \\ x(\delta(v)) &\le 1 && v \in V \\ x &\ge 0 && e \in E \end{aligned}$$

17.1.1 Optimal Integral Solution

Reduction

Let x be an optimal basic solution.

Suppose there is an edge e such that $x_e = 0$. Delete it and the result follows by induction.

Suppose now there is an edge uv with $x_{uv} = 1$. Recurse on $G - \{u, v\}$ to get an integral solution corresponding to a matching M' which attains $\mathrm{OPT} - c_{uv}$. Add e to M' and the result follows.

Rank Argument

Lemma 17.1.1
There exists an edge e with $x_e \in \{0, 1\}$ in ANY basic solution.

Proof
Suppose for a contradiction that $0 < x < 1$.

Recall that there are $|E|$ linear independent tight constraints since x is basic. Let W be the set of eight degree constraints.

Clearly the non-negativity constraints are NOT tight. Every $v \in W$ has degree two or else the edge variables incident to v do not add to 1.

We have

$$\begin{aligned}|W| &\geq |E| \\ &= \frac{1}{2}\left(\sum_{v \in W} \deg v + \sum_{v \notin W} \deg v\right) \\ &\geq \frac{1}{2} 2|W| \\ &= |W|\end{aligned}$$

In particular, we have equality throughout.

This means that $|W| = |E|$ and $\deg(v) = 2$ for all $v \in W$. Moreover, $\deg(v) = 0$ for $v \notin W$.

We claim that there at most $|W| - 1 = |E| - 1$ tight degree constraints which are also linearly independent. This would contradict the fact that x is a basic solution.

Notice the degree constraint for $v \in V$ is just

$$\langle \chi_{\delta(v)}, x \rangle \leq 1.$$

Let $V = (A, B)$ be a bipartition of G.

$$\sum_{v \in W \cap A} \chi_{\delta(v)} - \sum_{v \in W \cap B} \chi_{\delta(v)} = 0$$

since we are essentially just counting each edge twice. Note we used the fact that $\deg(v) = 0$ for all $v \notin W$.

17.2 3-Dimensional Matching

Problem 12 (3-Dimensional Matching)
Given a tripartite hypergraph

$$G = (X, Y, Z)$$

find a maximum (weight/cardinality) matching.

The LP relaxation is still

$$\begin{aligned} \max\langle 1_m, x\rangle & \\ x(\delta(v)) \le 1 & \qquad v \in V \\ x_e \ge 0 & \qquad e \in E \end{aligned}$$

17.2.1 Basic Solution

Lemma 17.2.1
For any basic solution $x > 0$, there is a hyperedge e with $x_e \ge \frac{1}{2}$.

Proof
Let W be the set of tight degree constraints.

Since $x_e < \frac{1}{2}$, each vertex $v \in W$ has degree at least 3 (or else $x(\delta(v)) < 1$).

We have

$$\begin{aligned} |W| &\ge |E| \\ &= \frac{1}{3}\left(\sum_{v\in W} 3 + \sum_{v\notin W} 0\right) \\ &= |W|. \end{aligned}$$

Thus $|W| = |E|$, $\deg(v) = 3$ for all $v \in W$, and $\deg(w) = 0$ for all $v \notin W$.

In particular, each hyperedge intersects

$$W \cap X, W \cap Y, W \cap Z$$

exactly once. Thus the constraints in W are linearly dependent.

This contradicts the assumption that x is a basic solution.

17.2.2 Iterative Rounding Algorithm

White $E \neq \varnothing$,

1) compute an optimal basic solution x
2) if there is a hyperedge e with $x_e = 0$, delete e
3) otherwise there is an edge $e = uvw$ with $x_e \geq \frac{1}{2}$, add e to the solution an recurse on $G - \{u, v, w\}$

Return the solution.

17.2.3 Approximation Guarantee

If we remove a hyperedge e with $x_e = 0$, then by induction the smaller hypergraph as an integral solution at least $\frac{1}{2}$ of the optimal LP solution.

Focus on the case when we add the hyperedge e with $x_e \geq \frac{1}{2}$. Then

$$\begin{aligned} \mathrm{OPT}(G - \{u, v, w\}) &= \mathrm{OPT}(G) - x_e - 3(1 - x_e) && \text{deleted } u, v, w \\ &= \mathrm{OPT}(G) - 3 + 2x_e \\ &\geq OPT(G) - 2. \end{aligned}$$

By induction there is an integral solution attaining at least

$$\frac{\mathrm{OPT}(G) - 2}{2} = \frac{\mathrm{OPT}(G)}{2} - 1.$$

Adding $c_e = 1$ yields that $\frac{1}{2}$-approximation.

17.3 General Assignment Question

Problem 13 (General Assignment Problem)
There are m machines $M_1, \dots, M_m$ and n jobs $J_1, \dots, J_n$.
There is a total available time T_i on each machine M_i.
If we assign job j to machine i, the cost is c_{ij} and the processing time is p_{ij}.
Assign each job to a machine and minimize the total cost while satisfying the time constraints.

17.3.1 LP Relaxation

Let x_{ij} be a variable indicating that job j is assigned to machine i. Let E be the set of all possible pairs i, j. Initially, every pair is possible, but we will delete pairs.

$$\begin{aligned}
&\min \sum_{i\in M, j\in J} c_{ij}x_{ij} && \\
&\sum_{i\in M, ij\in E} x_{ij} = 1 && j \in J \\
&\sum_{j\in J: ij\in E} p_{ij}x_{ij} \leq T_i && i \in M \\
&x_{ij} \geq 0 && ij \in E
\end{aligned}$$

Remove all pairs i, j with $p_{ij} > T_i$. These are never in the optimal solution.

17.3.2 Iterative Relaxation Algorithm

While $J \neq \varnothing$

1) Compute an optimal basic solution x. If infeasible, return “impossible”
2) Delete all pairs i, j with $x_{ij} = 0$ from E
3) If $x_{ij} = 1$, assign job j to machine i. Update $T_i := T_i - p_{ij}$ and $J := J - j$
4) If there is a machine i with degree 1, remove the time constraint from machine i
5) If there is a machine i with degree 2 and fractional degree at least 1, remove the time constraint from machine i

The new idea is to remove constraints.

17.3.3 Performance Guarantee

First we assume the algorithm always terminates and prove the approximation guarantee.

Since we only assign a job j to a machine if $x_{ij} = 1$, the cost is no more than the optimal cost (relaxing constraints only decreases optimal cost).

It remains to consider the time violation.

The case when there is a machine i with degree 1 clearly violates the time constraint by at most T_i.

Focus on the case for the machine i with degree 2 (say jobs j, k) and fractional at least 1. In the worst case, both jobs are assigned to machine i. Then the time violation is at most

$$\begin{aligned}(1 - x_{ij})p_{ij} + (1 - x_{ik})p_{ik} &\leq (2 - x_{ij} - x_{ik})T_i \\ &\leq T_i\end{aligned}$$

Theorem 17.3.1
Suppose there is an assignment with total cost C satisfying all time constraints. There is a polynomial time algorithm to find an assignment with total cost C. Moreover, the i-th time constraint is violated by at most T_i.

It is NP-hard just to find an assignment satisfying all time constraints.

17.3.4 Basic Solution

It remains to show that one of the 4 cases MUST apply to a basic solution x.o

Suppose $0 < x < 1$. Let J^* be the tight job constraints and W^* be the tight machine constraints.

Due to $x < 1$, $\deg(j) \geq 2$ for each $j \in J^*$. If there is no machine with $\deg(i) = 1$, then $\deg(i) \geq 2$ for all machines $i \in M^*$.

Then

$$\begin{aligned}|J^*| + |M^*| &\geq |E| \\ &\geq \frac{1}{2}\left(\sum_{e \in M^*} 2 + \sum_{j \in J^*} 2\right) \\ &\geq |J^*| + |M^*|\end{aligned}$$

The equalities imply $\deg(i) = 2$ for all $i \in M^*, j \in J^*$ and $\deg(i) = 0$ for all $i \notin M^*$.

Moreover, $\deg(j) = 2$ for all $j \in J^*$ and $\deg(j) = 0$ for all $j \notin J^*$.

Now, the sum of fractional degrees for each tight job is at least 1, since each job must be assigned (albeit fractionally) to one machine. Thus there MUST be a machine with fractional degree at least 1. This shows that the last case of the algorithm occurs.

17.4 General Matchings

17.4.1 Edmonds' LP

Edmonds gave an exponential-sized LP and proved its integrality.

$$\begin{aligned} &\max\langle c, x\rangle && \\ x(\delta(v)) &\leq 1 && v \in V \\ x(E(S)) &\leq \frac{|S|-1}{2} && S \subseteq V, |S| \text{ is odd} \\ x_e &\geq 0 && e \in E \end{aligned}$$

There is a polynomial time separation oracle for this LP, but it is quite nontrivial. Specifically, general matchings can be solved in polynomial time with the ellipsoid method.

Chapter 18

Spanning Tree Polytopes

18.1 Spanning Trees

Consider the following LP for the minimum spanning tree problem

$$\begin{aligned} &\max\langle c, x\rangle \\ &x(E(S)) \leq |S| - 1 && \forall S \subset V \\ &x(E(V)) = |V| - 1 \\ &x_e \geq 0 \end{aligned}$$

There is a polynomial time separation oracle using maximum-flow minimum-cut techniques. Thus the ellipsoid method solves this (exponential sized) LP in polynomial time.

18.1.1 Rank Argument

We want to show this LP is integral. There are many constraints, but surprisingly few linearly independent ones.

Basic Tight Constraints

Definition 18.1.1 (Tight Set)
S is a tight set if

$$x(E(S)) = |S| - 1.$$

For $F \subseteq E$, let χ_F be the characteristic vector. Remark that $\chi_{E(S)}$ is the row vector in the CONSTRAINT matrix for the tight set S.

Laminar Family

To show that a rank is small, we show there is a basis with special structure.

We say S, T are intersecting if

$$S - T, T - S, S \cap T$$

are all non-empty.

Definition 18.1.2 (Laminar)
A family $\mathcal{L}$ of sets is a laminar family if there are no pairwise intersecting sets.

Proposition 18.1.1
Let $\mathcal{L}$ be a laminar family on a ground set of n elements. Then

$$|\mathcal{L}| \leq 2n - 1.$$

Furthermore, if $\mathcal{L}$ has no singleton sets, then

$$|\mathcal{L}| \leq n - 1.$$

Proof
By induction.

Lemma 18.1.2
For $S, T \subseteq V$

$$\chi_{E(S)} + \chi_{E(T)} \leq \chi_{E(S \cap T)} + \chi_{E(S \cup T)}.$$

Furthermore, equality holds if and only if

$$E(S - T, T - S) = \varnothing$$

where $E(X, Y)$ denotes the set of edges with one endpoint in X and the other in Y.

Proof
By cases.

Uncrossing Technique

We argue that tight sets are closed under union and intersections. Furthermore, they are linearly independent. We can thus replace them by their union and intersection.

Lemma 18.1.3
If $S, T \in \mathcal{F}$ and $S \cap T \neq \varnothing$ then both $S \cap T, S \cup T$ are in $\mathcal{F}$.
Furthermore

$$\chi_{E(S)} + \chi_{E(T)} = \chi_{E(S\cap T)} + \chi_{E(S\cup T)}.$$

Proof
By tightness

$$\begin{aligned}
|S| - 1 + |T| - 1 &= x(E(S)) + x(E(T)) \\
&\leq x(E(S\cap T)) + x(E(S \cup T)) && (*) \\
&\leq |S \cap T| - 1 + |S \cup T| - 1 \qquad \text{not necessarily tight constraints} && (\star) \\
&= |S| - 1 + |T| - 1.
\end{aligned}$$

$(*)$ Since we removed all variables $x_e = 0$, the previous lemma implies that

$$\chi_{E(S)} + \chi_{E(T)} = \chi_{E(S\cap T)} + \chi_{E(S\cup T)}.$$

$(\star)$ Equality here implies

$$S \cup T, S \cap T \in \mathcal{F}$$

Laminar Basis

Theorem 18.1.4
Let

$$\mathcal{F} := \{S : x(E(S)) = |S| - 1\}$$

be the set of all tight sets.
There exists a laminar family $\mathcal{L} \subseteq \mathcal{F}$ such that

$$\operatorname{span} \mathcal{L} = \operatorname{span} \mathcal{F}.$$

We say S intersects R if $R - S, S - R \neq \varnothing$.

Proof
We argue that if $\mathcal{L}$ is a maximal laminar sub-family of $\mathcal{F}$ with respect to inclusion, then

$$\operatorname{span} \mathcal{L} = \operatorname{span} \mathcal{F}.$$

Suppose towards a contradiction this is false. Let $\mathcal{R} \subseteq \mathcal{F}$ be the set beyond $\operatorname{span} \mathcal{L}$.

Observe that no set $F \in \mathcal{F}$ is such that $\mathcal{L} \cup \{F\}$ is a laminar family by maximality. Then,

> let $R \in \mathcal{R}$ be such that it intersects the minimal number of sets in $\mathcal{L}$. Say $S \cap R \neq \varnothing$ for some $S \in \mathcal{L}$. We claim that $S \cap R, S \cup R$ both intersect fewer sets of $\mathcal{L}$ than R.
>
> Assuming this claim, we know by the previous lemma that $S \cap R, S \cup R$ both reside in $\mathcal{F}$. But they cannot be in $\mathcal{R}$ by the minimality of R. Thus they both reside in $\operatorname{span} \mathcal{L}$ and we can express
>
> $$\chi_{E(R)} = \chi_{E(S\cap R)} + \chi_{(S\cup R)} - \chi_{E(S)} \in \operatorname{span} \mathcal{L}.$$
>
> This is the desired contradiction.
>
> It remains to show the claim.
>
> It is clear the intersecting number of $S \cap R$ is at most the intersecting number of R. But $S \cap R \subseteq S$ so while R intersects S, $S \cap R$ does not. Thus the inequality is strict.
>
> Similarly, as $\mathcal{L}$ is a laminar family, the intersecting number of $S \cup R$ is at most that of R. But $S \cup R$ again does not intersect S. Thus this inequality is also strict.

Rank Argument

Theorem 18.1.5
The number of linearly independent subtour elimination constraints is at most $|V| - 1$.

> **Proof**
> There exists a laminar basis. This has size at most $n - 1$.

Integrality

Let x be a basic solution and remove edge e where $x_e = 0$. Notice the subtour elimination constraints implies $x \leq 1$.

By the theorem, a basic solution has at most $n - 1$ tight constraints and therefore at most $n - 1$ non-zero edge variables. By the second constraint, the sum of edge variables is $n - 1$, thus there is exactly $n - 1$ integral edge variables.

18.1.2 Submodular Functions

Definition 18.1.3 (Submodular Function)
A function $f : 2^V \to \mathbb{R}$ is submodular if it satisfies

$$f(S) + f(T) \geq f(S \cup T) + f(S \cap T)$$

for all $S, T \subseteq V$.

One important example is the cut function. The function we considered before was actually supermodular (the inequality is reversed). However, we need only negate a supermodular function to achieve a submodular function.

Submodular functions are central toward many polynomial time solvable combinatorial optimization problems. This uncrossing technique is used frequently to deal with submodular functions.

18.2 Minimum Bounded Degree Spanning Tree

Problem 14 (Minimum Bounded Degree Spanning Tree)
Given an undirected graph $G = (V, E)$, a cost c_e on each $e \in E$, and a degree upper bound B_v on each $v \in V$, find a minimum cost spanning tree satisfying all the degree upper bounds.

This problem is clearly NP-hard, since $B = 2$ is the Hamiltonian path problem.

Theorem 18.2.1
If there is a spanning tree with cost C satisfying all the degree constraints, there is a polynomial time algorithm to find a spanning tree T with cost at most C and $\deg_T(v) \leq B_v + 1$ for all $v \in V$.

Corollary 18.2.1.1
There is a polynomial time algorithm to find a minimum spanning tree with maximum degree at most

$$\mathrm{OPT} + 1.$$

18.2.1 LP Relaxation

We add an additional constraint to the spanning tree LP.

$$
\begin{aligned}
\max\langle c, x\rangle & & \\
x(E(S)) &\leq |S| - 1 & \forall S \subset V \\
x(E(V)) &= |V| - 1 & \\
x(\delta(v)) &\leq B_v & v \in W \\
x_e &\geq 0 &
\end{aligned}
$$

where W is the set of vertices with degree constraints.

18.2.2 Iterative Relaxation

We show an algorithm which violates the degree bound by at most 2.

Initialize $F = \varnothing$. It is the partial solution we have seen so far.

While $|V| \geq 2$

1) Compute an optimal basic solution x. If it is infeasible, return "impossible"
2) Remove all edges with $x_e = 0$ from E
3) If there is a vertex v with only one edge uv incident to it, then set $F := F \cup \{uv\}$, $V := V - v$, and $B_u := B_u - 1$
4) If there is a vertex v with $\deg(v) \leq 3$, remove the degree constraint on v (ie $W := W - v$)

Return F.

In essence, we iteratively identifies a leaf of the spanning tree and remove it from the graph.

18.2.3 Approximation Guarantee

First, let us assume the algorithm terminates correctly and prove its correctness.

Consider the case where there is a degree 1 vertex incident to $e \in E$. We claim $x_e = 1$. Indeed, by the subtour elimination constraints

$$x(E(V - v)) \leq |V| - 2$$

while

$$x(E(V)) = |V| - 1$$

thus the sum of edge variables in the cut $\delta(v)$ is 1. But there is only one edge, so $x_e = 1$.

Note here that the cost of consuming this edge has been taken into account when computing the optimal basic solution. Therefore, we never output a tree with more than optimal weight.

Then consider the case when there are no leaf vertices but there is a vertex v of degree at most 3. In the worst case, all edge incident to v will be chosen. Observe that the degree constraint is at least 1, or else no spanning tree exists for any non-trivial graphs. But then we only violate the degree constraint by at most 2.

18.2.4 Analysis of Basic Solution

We now prove termination. The hard work has already been done.

We want to show that at every iteration, the current graph falls into one of our 3 cases. Indeed, suppose that we have removed all edges e where $x_e = 0$. Moreover, suppose the minimum degree is at least 2.

We should be in the final case. Suppose towards a contradiction that there are no vertices with degree constraints that have degree at most 3.

Let W be the set of vertices with degree constraints.

$$\begin{aligned}|E| &= \frac{1}{2}\left(\sum_{v\in W}\deg(v) + \sum_{v\notin W}\deg(v)\right)\\ &\geq \frac{1}{2}\left(\sum_{v\in W}4 + \sum_{v\notin W}2\right)\\ &= |V| + |W|.\end{aligned}$$

On the other hand, having removed the zero edge variables, the number of edges $|E|$ are at most the number of linearly independent constraints. In particular

$$|E| \leq |V| - 1 + |W|.$$

Here we used our result from the previous section about linearly independent subtour constraints.

We have the desired contradiction.

Further Optimizations

A careful analysis which takes into account laminarity reveals that the degree violation can be decreased to at most 1.

While $W \neq \varnothing$

1) Compute an optimal basic solution x. Remove all edge with $x_e = 0$

2) Remove the degree constraint on a vertex v with $\deg_G(v) \leq B_v + 1$.

Return a minimum spanning tree.

Chapter 19

Linear Programming Duality

19.1 Linear Programming Duality

19.1.1 Motivation

Consider the maximization problem

$$\max\langle c, x\rangle$$

subject to

$$\langle a_i, x\rangle \leq b_i$$

where a_i is the i-th row of the constraint matrix.

By taking non-negative linear combinations of the constraints, we can obtain an upper bound on the objective value. Suppose

$$\sum_{i=1}^{m} a_i y_i \geq c$$

then

$$\sum_{i=1}^{m} y_i b_i \geq \sum_{i=1}^{m} y_i \langle a_i, x\rangle \geq \langle c, x\rangle.$$

If we wish to find the best coefficients, this is a linear program!

19.1.2 Weak Duality

The dual LP is defined as

$$\max\langle b, y\rangle, y^T A \geq c, y \geq 0.$$

Theorem 19.1.1 (Weak Duality)
For any feasible x and feasible y in the primal/dual programs

$$\langle c, x\rangle \leq \langle b, y\rangle.$$

19.1.3 Complementary Slackness Conditions

We derive necessary and sufficient conditions to achieve primal dual equality.

Theorem 19.1.2 (Complementary Slackness)
If x, y are optimal in the primal/dual if and only if

$$\begin{aligned} x_j > 0 &\implies \langle a^j, y\rangle = c_j \\ y_i > 0 &\implies \langle a_i, x\rangle = b_i \end{aligned}$$

Here superscripts denote columns and subscripts denote rows.

Proof
Use the inequalities

$$y^T b \geq y^T Ax \geq c^T x.$$

19.1.4 Primal-Dual Algorithms

The idea is to directly find feasible solutions satisfying the complementary slackness conditions. One example is the Hungarian matching algorithm.

19.1.5 Strong Duality

Separation Theorem

Theorem 19.1.3 (Separation)
Let $S \subseteq \mathbb{R}^n$ be a closed convex set and $v \notin S$.
There exists $w \in \mathbb{R}^n$ such that

$$\langle w, v\rangle > \langle w, x\rangle$$

for all $x \in S$.

Proof
First show there is a closest point using compactness and continuity. Then observe that the averge of two closest points is strictly better thus the closest point is unique.

Say $v^* \in S$ is the closest point, take

$$w := v - v^*.$$

Farkas' Lemma

Lemma 19.1.4 (Farkas)
The system $Ax = b$ for $x \geq 0$ has no solutions if and only if there is y such that

$$y^T A \geq 0$$

and

$$y^T b < 0.$$

Proof
By contradiction and the separation theorem.

Strong Duality

Theorem 19.1.5
If the primal and dual programs are feasible, they have the same objective value.

Proof
Show using Farkas' lemma that if there is μ which is strictly greater than the optimal objective value of the primal, then μ is strictly greater than the optimal objective value of the dual.

The idea is to reduce the optimization problem to a decision problem concerning feasibility.

19.2 Min-Max Theorems

Here are some powerful results in combinatorial optimization.

All results use total unimodularity of the constraint matrices and strong duality.

19.2.1 Bipartite Matching & Vertex Cover

The maximum bipartite matching is the same size as the minimum vertex cover.

19.2.2 Maximum-Flow Minimum-Cut

The maximum *st*-flow is equivalent to the minimum *st*-cut.

19.3 Game Theory

A two-player zero-sum game can be described by a matrix, where each row corresponds to a strategy and the column corresponds to a strategy of the column player. The row player choses strategy i and the column player chooses strategy j and then the payoff is the i, j-th row of the matrix.

The row player wishes to maximize payoff, while the column player wishes to minimize the payoff. A Nash equilibrium is a pair of strategies such that even is a player knows the strategy of the other player, they cannot gain by changing their strategy.

19.3.1 Minimax Theorem

Let $A \in \mathbb{R}^{m\times n}$ payoff matrix and $x \in \Delta^m, y \in \Delta^n$ be probability distributions.

The expected payoff of these strategies is

$$x^T Ay.$$

Theorem 19.3.1 (Von-Neumann Minimax)
We have

$$\max_{x\in\Delta^m} \min_{y\in\Delta^n} x^T Ay = \min_{y\in\Delta^n} \max_{x\in\Delta^m} x^T Ay.$$

Proof
Strong duality.

19.3.2 Yao's Minimax Principle

We can consider a randomized algorithm as a distribution of deterministic algorithms. Think of it as a two-player zero-sum game where the algorithm plays the best distribution of deterministic algorithms to minimize running time while the adversary chooses the best distrubution of inputs to maximize the running time.

Using Von-Neumann's theorem, it suffices to find an input such that ANY deterministic algorith takes a long time to solve it.

Chapter 20

Multiplicative Weight Update Method

20.1 Online Expert Model

20.1.1 Online Decisions

Suppose each day we decide if we want to buy or sell stocks based on if the stock market falls or gains. The correct prediction gives us a gain of \$1 and a false prediction costs \$1. There are n experts in the newspapers who make daily predictions. Given the track records of the experts, can we do as well as the best expert?

Problem 15 (Online Expert)
There is a set of n decisions for each of T rounds. After each round $t \in [T]$, we learn the outcome as a cost vector $m_t \in \mathbb{R}^n$.
We assume that

$$-1_n \leq m_t \leq 1_n.$$

In each round, we make a decision by specifying a probability distribution p_t over the n experts.
We wish to minimize the total cost comparing to the best expert. Here the total cost of expert i is

$$\sum_{t=1}^{T} m_t(i).$$

The total cost of our decision is

$$\sum_{t=1}^{T} \langle m_t, p_t \rangle.$$

20.1.2 Multiplicative Weight Update Method

We maintain a weight $w_t(i)$ on each expert representing our "trust" of that expert at time t. Initially set

$$w_1 := 1_n.$$

Fix a parameter $\eta \leq \frac{1}{2}$.

For each $1 \leq t \leq T$

1) Let $\Phi_t := \sum_{i=1}^n w_t(i)$
2) Set $p_t(i) := \frac{w_t(i)}{\Phi_t}$
3) Observe m_t
4) Update $w_{t+1}(i) := (1 - \eta m_t(i)) \cdot w_t(i)$

Analysis

Theorem 20.1.1 (Regret Minimization)
Assume $m_t(i) \in [-1, 1]$ and $\eta \leq \frac{1}{2}$. After T rounds

$$\sum_{t=1}^T \langle m_t, p_t \rangle \leq \sum_{t=1}^T m_t(i) + \eta \sum_{t=1}^T |m_t(i)| + \frac{\ln n}{\eta}$$

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

Lemma 20.1.2
The inequality

$$1 - \eta x \geq \begin{cases} (1-\eta)^x, & x \in [0, 1] \\ (1+\eta)^{-x}, & x \in [-1, 0] \end{cases}$$

holds.

Proof

Upper Bound on Φ_t By computation

$$\begin{aligned}
\Phi_{t+1} &:= \sum_{i=1}^n w_{t+1}(i) \\
&= \sum_{i=1}^n (1-\eta m_t(i))w_t(i) \\
&= \sum_{i=1}^n w_t(i) - \eta \sum_{i=1}^n m_t(i))w_t(i) \\
&= \sum_{i=1}^n w_t(i) - \eta \Phi_t \sum_{i=1}^n m_t(i))p_t(i) && p_t(i) := \frac{w_t(i)}{\Phi_t} \\
&= \Phi_t(1-\eta\langle m_t, p_t\rangle) \\
&\leq \Phi_t e^{-\eta\langle m_t, p_t\rangle} && 1-x \leq e^{-x} \\
&\leq \dots \\
&\leq \Phi_1 e^{-\eta \sum_{i=j}^t \langle m_j, p_j\rangle} \\
&= ne^{-\eta \sum_{i=j}^t \langle m_j, p_j\rangle}.
\end{aligned}$$

Lower Bound on Φ_t Again by computation

$$\begin{aligned}
\Phi_{t+1} &:= \sum_{i=1}^n w_{t+1}(i) \\
&\geq w_{t+1}(i) \\
&= 1 \cdot \prod_{j=1}^t (1-\eta m_j(i)) \\
&\geq (1-\eta)^{\sum_{\geq 0} m_t(i)} (1+\eta)^{-\sum_{<0} m_t(i)}.
\end{aligned}$$

Remark now by the choice of η that

$$\begin{aligned}
\ln(1-\eta) &\geq -\eta - \eta^2 \\
\ln(1+\eta) &\geq \eta - \frac{\eta^2}{2} \\
&\geq \eta - \eta^2.
\end{aligned}$$

This can be see from the Taylor Expansion of $\ln(1+x)$.

> Taking logs from both sides of the bounds, we see that
>
> $$\begin{aligned} \ln n - \eta \sum_{t=1}^{T} \langle m_t, p_t \rangle &\geq \sum_{\geq 0} m_t(i) \ln(1-\eta) - \sum_{<0} m_t(i) \ln(1+\eta) \\ &\geq \sum_{\geq 0} m_t(i)(-\eta - \eta^2) - \sum_{<0} m_t(\eta - \eta^2) \\ &= -\eta \sum_{i=1}^{n} m_t(i) - \eta^2 \sum_{i=1}^{n} |m_t(i)|. \end{aligned}$$
>
> Rearranging the inequality yields the result.

Notice that the assumption that $m_t(i) \in [-1, 1]$ was important. This bounds the mistake we make in each round. We can rescale $m_t(i) \in [-w, w]$ to the case we proved but the additive error is then

$$\frac{w \ln n}{\eta}.$$

The parameter w is called the width, and is crucial in the analysis of multiplicative update methods.

20.2 Solving Linear Programs

Suppose we have an LP of the form

$$\min\langle c, x \rangle, Ax \geq b, x \geq 0.$$

To solve this, let us reduce this to a feasibility problem

$$Ax \geq b, x \geq 0.$$

Instead of finding a solution which satisfies m constraints simultaneously, it would be easier to satisfy a convex of the constraints

$$p_t Ax \geq p_t b, x \geq 0.$$

If this problem is infeasible, then the original LP is infeasible. If it is feasible, then as least one constraint is satisfied.

We use the MWU method to put more weight on constraints that are violated to obtain p_{t+1} and repeat the procedure. After T iterations for a given T, we return the average solution

$$\bar{x} := \frac{1}{T} \sum_{t=1}^{n} x_t$$

as our approximation solution.

20.2.1 Feasibility

How do we find a solution satisfying

$$p_t Ax \geq p_t b$$

such that $x \geq 0$?

If $p_t A$ has a positive entry, we can use that entry in x to obtains $p_t b$. Otherwise we would have

$$p_t A \leq 0, p_t b > 0$$

and p_t is a certificate of infeasibility.

20.2.2 Oracle & Width

Given a constraint $p_t Ax \geq p_t b$, we assume that there is an oracle which returns a solution x satisfying

$$p_t Ax \geq p_t b, x_t \geq 0.$$

We say an oracle is of width w if the returned solution x_t always satisfies

$$\left| A(i)^T x_t - b_i \right| \leq w$$

for $1 \leq i \leq m$. The runtime of the method is heavily dependent on the width, as it is the loss bound for

$$m_t(i) \in [-w, w]$$

within the online expert problem.

We will now proceed assuming there is an oracle of width w to solve $p_t Ax \geq p_t b$ subject to $x_t \geq 0$.

20.2.3 The Algorithm

Initially

$$p_1(i) = \frac{1}{m}$$

for each $1 \leq i \leq m$.

For $1 \leq t \leq T$

1) Apply the oracle to find a solution x_t satisfying $p_t Ax_t \geq p_t b$ and $x_t \geq 0$ with $\left| a_i^T x_t - b_i \right| \leq w$ for each $1 \leq i \leq m$

2) Set the outcome $m_t(i) := \frac{a_i^T x_t - b_i}{w}$ for $1 \leq i \leq m$

3) Use $m_t(i)$ to apply the multiplicative weight update method to compute p_{t+1}

Return $\bar{x} := \sum_{t=1}^{T} \frac{x_t}{T}$.

As a reminder, the multiplicative weights update method sets

$$\begin{aligned} w_{t+1}(i) &:= (1 - \eta m_t(i)) w_t(i) \\ p_{t+1}(i) &:= \frac{w_{t+1}(i)}{\Phi_{t+1}} \end{aligned}$$

where $\Phi_{t+1} = \sum w_t(i)$ and $w_1 := 1_m$ initially.

20.2.4 Performance Guarantee

Theorem 20.2.1
Given $\epsilon > 0$ and an oracle of width w, there is an algorithm which either finds a solution $\bar{x}$ such that

$$a_i^T \bar{x} \geq b_i - \epsilon$$

for $1 \leq i \leq m$ or concludes that the LP is infeasible.
The algorithm calls the oracle at most

$$O\left(\frac{w^2 \log m}{\epsilon^2}\right)$$

times, with an additional $O(m+n)$ time per call. With each call, there is an additional $O(m+n)$ time bookkeeping involved.

Proof
Suppose the algorithm does not find a certificate of infeasibility. The regret minimization theorem says

$$\begin{aligned} \sum_{t=1}^{T} \left(\frac{a_i^T x_t - b_i}{w}\right) + \eta T + \frac{\ln m}{\eta} &\geq 0 \\ \sum_{t=1}^{T} \left(\frac{a_i^T x_t - b_i}{T}\right) &\geq -\eta w - \frac{w \ln m}{\eta T}. \end{aligned}$$

> An appropriate choice of η, T yields the bound. Specifically,
>
> $$\eta \in O\left(\frac{\epsilon}{2w}\right)$$
> $$T \in O\left(\frac{w^2 \ln m}{\epsilon^2}\right).$$

20.2.5 Remarks

An oracle of small width is crucial for the time complexity of the method. For combinatorial problems, we usually have a straightforward bound that $w \in O(n)$.

Notice that the ellipsoid and interior point method has time complexity dependent on $\log \epsilon^{-1}$ while the MWU method has dependency ϵ^{-1}. This means it is less competitive.

Chapter 21

Laplacian Solvers

21.1 Maximum Flow

21.1.1 Undirected Graphs

Problem 16 (Maximum Flow)
Given an undirected graph $G = (V, E)$ where each edge e has capacity c_e and two vertices s, t, find a maximum st-flow subject to capacity and flow conservation constraints.

The LP relaxation is

$$\begin{aligned}
&\max f(\delta(s)) - f(\delta(\bar{s})) && \\
f(\delta(v)) &= f(\delta(\bar{s})) && v \in V - s - t \\
f_e &\leq c_e && e \in E \\
f_e &\geq 0 && e \in E
\end{aligned}$$

Through binary search, we can reduce this to a feasibility problem.

21.1.2 Multiplicative Weight Update Method

Consider the following feasibility version of maximum flow.

$$\begin{aligned}
f(\delta(s)) &= k + f(\delta(\bar{s})) && \\
f(\delta(v)) &= f(\delta(\bar{s})) && v \in V - s - t \\
f_e &\le c_e && e \in E \\
f_e &\ge 0 && e \in E
\end{aligned}$$

Let us group the first 3 sets of constraints as "easy" constraints and put the capacity constraints as "hard" constraints. We use the MWU method to combine the hard constraints into one.

The idea is to design an oracle to return a solution f satisfying the average constraint, as well as the easy constraints simultaneously. We would like the oracle to be small. Namely $f \le p$ for a small p.

21.1.3 Electric Flow Oracle

We want an oracle satisfying the following properties. Given the weights w_e and $\epsilon > 0$, return $f \in \mathbb{R}^m$ where

(i) f satisfies flow conservation, non-negative constraints, and objective value constraint

(ii) If the problem is feasible, f satsifies $\sum w_e f_e \le (1+\epsilon) \sum w_e$

(iii) When the problem is feasible, f satisfies $f_e \in O\left(\sqrt{\frac{m}{\epsilon}}\right)$

(iv) The runtime of the oracle is $\tilde{O}(m)$

Given w_e and $\epsilon > 0$, put $W := \sum w_e$. Set the resistance of each $e \in E$ to be

$$r_e := w_e + \frac{\epsilon W}{m}.$$

Return the electrical flow f sending k units of flow $s \to t$.

Analysis

Property (i) follows directly from the definition of electric flow. On the other hand, property (iv) is a breakthrough result we will see soon.

Proof (Property (ii))
Recall that the energy of flow f is

$$\sum f_e^2 r_e$$

and the electrical flow minimizes the energy among all possible flows.

Let f^* be a k-unit st-flow such that

$$f^* \leq 1.$$

Thus

$$\begin{aligned} \varepsilon(f) &\leq \varepsilon(f^*) \\ &= \sum (f_e^*)^2 r_e \\ &\leq \sum r_e \\ &= \sum w_e + \frac{\epsilon W}{m} \\ &= (1+\epsilon)W. \end{aligned}$$

By the Cauchy-Schwartz inequality

$$\begin{aligned} \left(\sum w_e f_e\right)^2 &\leq \left(\sum w_e f_e^2\right)\left(\sum w_e\right) \\ &\leq \left(\sum r_e f_e^2\right) \cdot W \\ &\leq (1+\epsilon)W^2 \end{aligned}$$

Taking square roots yields the desired inequality.

Proof (Property (iii))
By computation

$$\begin{aligned} f_e^2 \left(\frac{\epsilon W}{m}\right) &\leq f_e^2 r_e \\ &\leq \varepsilon(f) \\ &\leq (1+\epsilon)W \end{aligned}$$

$$\begin{aligned} f_e &\leq \sqrt{\frac{(1+\epsilon)m}{\epsilon}} \\ &\in O\left(\sqrt{\frac{m}{\epsilon}}\right). \end{aligned}$$

21.1.4 Performance Guarantee

Given such an oracle exists, the MWU method implies the calling the oracle

$$O\left(\frac{\sqrt{m}\log n}{\epsilon^{2.5}}\right)$$

times, the average solution satisfies all the constraints

$$f_e \leq 1 + O(\epsilon).$$

Initially, every edge has the same resistance and we send a k-unit electrical flow $s \to t$. Some capacity constraints may be violated. In each iteration, update the weight

$$w_{t+1}(e) := w_t(e) \cdot \left(1 + \frac{\epsilon}{p} f_t(e)\right).$$

Intuitively, if the flow on an edge exceeds capacity, increase the resistance on that edge.

By scaling

$$f_e \mapsto \frac{f_e}{1 + O(\epsilon)},$$

this flow will satisfy all constraints and has objective value

$$\frac{k}{1 + O(\epsilon)}.$$

To conclude, we have a

$$\tilde{O}\left(\frac{m^{\frac{3}{2}}}{\epsilon^{2.5}}\right)$$

time algorithm to return a $(1 - O(\epsilon))$-approximate solution.

21.1.5 Remarks

The performance guarantee from MWU has a factor of w^2. If we have the non-negative guarantee, then this improves to w. We used this in our work.

We only required that our oracle approximately satisfies the convex combination of constraints. Since the MWU only finds an approximately feasible solution anyways, this is not an issue.

We can "round" the fractional solution to an integral solution in near-linear time. Thus if we have a fractional solution, this implies an integral one.

It is possible to obtain a

$$\tilde{O}\left(\frac{m^{\frac{4}{3}}}{\epsilon^3}\right)$$

time algorithm to return a $(1 - O(\epsilon))$-approximate solution by removing high congestion edges.

This has been improved to a run time of

$$\tilde{O}\left(\frac{m}{\epsilon^3}\right).$$

Recently, a

$$\tilde{O}(m^{\frac{4}{3}})$$

time exact algorithm has been found for directed unweighted graphs, involving interior point methods. An important subroutine of this method is the near linear time Laplacian solver.

21.2 Laplacian Solvers

The original paper has run time

$$O(m \log^c n)$$

where c was pretty big. This has been improved to

$$\tilde{O}(m\sqrt{\log n})$$

where we hide some $\log \log n$ terms.

We briefly mention some simpler, more elegant algorithms.

21.2.1 Electrical Flow Perspective

We start with an arbitrary flow satisfying the demands, and slowly change it to the electrical flow.

This involves "fixing" "bad" cycles repeatedly. An efficient implementation fixes only "fundamental cycles" with respect to a tree T. The fastest version requires "low stretch spanning trees" and random sampling non-tree-edges to "fix" the containing cycle.

A further improvement involes data structures. The energy of the flow is the potential function. The analysis uses some primal-dual analysis.

Overall, this gives a

$$O\left(m \log^3 n \log \frac{1}{\epsilon}\right)$$

time algorithm to obtain a $(1+\epsilon)$-approximate solution.

21.2.2 Gaussian Elimination

We use Gaussian elimination to solve Laplacian equations. When we eliminate a vertex v, this corresponds to creating a complete graph on $N(v)$.

This makes the graph denser and denser. The work around is to sparsify the complete graph.

The analysis is nontrivial and requires a careful application of the martingale concentration inequalities.

Overall, this gives a

$$O\left(m \log^3 n \log \frac{1}{\epsilon}\right)$$

time algorithm to obtain a $(1+\epsilon)$-approximate solution.