Probabilistic Methods in Computer Science

Instructor: Seffi Naor
TA: Erez Petrank

Abstract

These notes summarize a course given at the Technion, in Spring semester, 1992, and again in Winter semester, 1993. The course focused on randomness in algorithms. It is comprised of three parts. The first one introduces the probabilistic method as a powerful tool for proving the existence of combinatorial objects. The techniques introduced are important for the rest of the course. The second part discusses several randomized algorithms in various settings. The third part completes the cycle and deals with methods for removing randomness from algorithms so as to make them deterministic. Homework exercises are attached. The book of Alon and Spencer, "The Probabilistic Method" [AS1], and the forthcoming book of Motwani and Raghavan, "Randomized Algorithms" [MR], were very helpful.

The material covered in the course is far from being inclusive. There are many areas in computer science where randomness is indispensable that were not discussed. A non-exhaustive list is: Cryptography, Number Theoretic algorithms, data structures, routing and parallel architectures, and weak sources of randomness. However, I believe that the techniques investigated here are essential for understanding the role of randomness.

The notes are the product of the students attending the course. The editing of the notes was completed while visiting the International Computer Science Institute at Berkeley and the Computer Science Department at Stanford University. Many thanks to: Yishay Ben-Aroya, Amir Ben-Dor, Alon Efrat, Roy Friedman, Shay Halevi, Yehuda Hershkovits, Yona Holländer, Michael Krivelevich, Sleiman Matar, Yael Petruschka, Asaf Shirazi, and Boaz Tamir.

Contents

Part I

The Probabilistic Method

Chapter 1

Basics of Probability Theory

1.1 Basic definitions

Definition 1.1.1 *A probability space is a triple* $(\Omega, F, \mathcal{P})$ *where*

- Ω *is a set.*
- F *is a family of subsets of* Ω *– The collections of* events *in the probability space.*
- $\mathcal{P}$ *is the probability measure, where* $p(\Omega) = 1$.

In this course we deal mainly with discrete spaces.

Definition 1.1.2 *We say that two events* A, B *are independent if*

$$\Pr[A \wedge B] = \Pr[A] \cdot \Pr[B]$$

.

Definition 1.1.3 *We say that two events* A, B *are disjoint if* $A \cap B = \phi$.

Claim 1.1.1 *If* $A \cap B = \phi$ *then* A *and* B *are not independent.*

Proof: $\Pr[A \cap B] = 0$ while $\Pr[A] \cdot \Pr[B]$ is not necessarily 0.

Definition 1.1.4 *A random variable X is a function $X : \Omega \longrightarrow R$ (from the elements of the space to the real numbers.) Let $p(\mathrm{x})$ denote* $\Pr[X = \mathrm{x}]$.

Definition 1.1.5 *The expectation of a random variable X, $E[X]$, is defined as $E[X] = \sum_x xp(x)$.*

The following is known as linearity of expectation.

Fact 1.1.1 *The expectation of the sum of two random variables X, Y is:*

$$E[X+Y] = E[X] + E[Y].$$

This can be extended to any finite number of random variables $X_1, \ldots, X_n$:

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

This can be further extended to any countable set of random variables $X_1, X_2, \ldots$
If $\sum_{i=1}^{\infty} E[|X_i|] < \infty$, then $\sum_{i=1}^{\infty} |X_i| < \infty$ almost surely, and:

$$E\left[\sum_{i=1}^{\infty} X_i\right] = \sum_{i=1}^{\infty} E[X_i]$$

Definition 1.1.6 *The variance of a random variable X is* $\mathrm{VAR}[X] = E\left[(X - E[X])^2\right]$.

Fact 1.1.2 $\mathrm{VAR}[X] = E[X^2] - (E[X])^2$.

Proof:

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

□

We denote the variance by σ^2 in the sequel, and call σ (the square root of the variance) the *standard deviation.*

Definition 1.1.7 *The k^{th} moment of a random variable X is defined to be $E\left[X^k\right]$.*

Please refer to [Pf, pp. 326-330] for more properties of expectations and higher moments.

1.2 The expectation and variance of the sum of random variables

Consider the random variables $X_1, X_2, \ldots, X_n$, where

$$X_i = \begin{cases} 0 & \text{with probability } p \\ 1 & \text{with probability } (1-p) = q \end{cases}$$

Let S_n be their sum: $S_n = \sum_{i=1}^{n} X_i$. According to Definition 1.1.5

$$E[X_i] = 0 \cdot p + 1 \cdot (1-p) = 1 - p = q$$

and

$$E\left[X_i^2\right] = 0^2 \cdot p + 1^2 \cdot q = q$$

Using Fact 1.1.1 we get that

$$E[S_n] = E\left[\sum_{i=1}^{n} X_i\right] = \sum_{i=1}^{n} E[X_i] = nq$$

$E[S_n^2] = E[(\sum X_i)^2] = E\left[\sum X_i^2 + \sum_{i \neq j} X_i \cdot X_j\right]$.

Again, from Fact 1.1.1 we get,

$$E\left[S_n^2\right] = \sum_{i=1}^{n} E\left[X_i^2\right] + \sum_{i \neq j} E[X_i \cdot X_j].$$

If X_i, X_j are pairwise independent, as in this case, $E[X_i \cdot X_j] = E[X_i] \cdot E[X_j]$. Hence,

$$\begin{aligned} E\left[S_n^2\right] &= nq + 2 \cdot \binom{n}{2} \cdot q^2 \\ &= nq + n(n-1)q^2 \\ &= nq(1-q) + n^2 q^2 \\ &= n^2 q^2 + npq \end{aligned}$$

Now we can use Fact 1.1.2 to derive that $\mathrm{VAR}[S_n] = npq + n^2q^2 - (nq)^2 = npq$.

1.3 Large deviation inequalities

1.3.1 Moment inequalities

Let X be a random variable and $\lambda > 0$.

Theorem 1.3.1 *The k^{th} moment inequality is:* $\Pr[|X| > \lambda] \leq \frac{E[|X|^k]}{\lambda^k}$.

Proof:

$$\begin{aligned}
E\left[|X|^k\right] &= \sum (|x|^k \cdot p(x)) \\
&= \sum_{|x|\le\lambda} (|x|^k \cdot p(x)) + \sum_{|x|>\lambda} (|x|^k \cdot p(x)) \\
&\ge \sum_{|x|>\lambda} (|x|^k \cdot p(x)) \\
&\ge \lambda^k \sum_{|x|>\lambda} p(x) \\
&= \lambda^k \Pr[|X| > \lambda].
\end{aligned}$$

In other words,

$$\Pr(|X| > \lambda) \le \frac{E\left[|X|^k\right]}{\lambda^k}$$

□

Some special cases:

1. For $k = 1$, if we replace λ for $\lambda \cdot E[X]$, we get *Markov's Inequality:*

$$\Pr[|X| \ge \lambda E[X]] \le \frac{1}{\lambda}.$$

2. For $k = 2$, if we replace $X - E[X]$ for X and $\lambda \cdot \sigma$ for λ, we get *Chebyshev's Inequality:*

$$\Pr[|X - E[X]| \ge \lambda \cdot \sigma] \le \frac{E[(X - E[X])^2]}{(\lambda\sigma)^2} = \frac{\mathrm{VAR}[X]}{\lambda^2 \mathrm{VAR}[X]} = \frac{1}{\lambda^2}.$$

Remark 1.3.1 *The proof of the k^{th} moment inequality holds for every non-decreasing function $g(x)$ on $[0, \infty)$:*

$$\Pr[|X| \ge \lambda] \le \frac{E[g(|X|)]}{g(\lambda)}.$$

1.3.2 Chernoff-type inequalities

Chernoff-type inequalities are derived by substituting the function e^{tx} as $g(x)$ in Remark 1.3.1. Hence, we get,

$$\Pr[|X| > \lambda] \le \frac{E\left[e^{tX}\right]}{e^{t\lambda}}$$

where t is chosen so as to minimize the right hand side.

For example, for $1 \le i \le n$, let the random variable X_i be

$$X_i = \begin{cases} 1 & \text{with probability } \frac{1}{2} \\ -1 & \text{with probability } \frac{1}{2} \end{cases}$$

We define $S_n = \sum_{i=1}^n X_i$.

Claim 1.3.1 $\Pr[|S_n| > \lambda] \le e^{-\frac{\lambda^2}{2n}}$.

Proof:

$$E\left[e^{tX_i}\right] = \frac{1}{2}(e^t + e^{-t}).$$

$$\begin{aligned} \frac{1}{2}(e^t + e^{-t}) &= \frac{1}{2}(1 + t + \frac{t^2}{2} + \frac{t^3}{6} + \ldots) + \frac{1}{2}(1 - t + \frac{t^2}{2} - \frac{t^3}{6} + \ldots) \\ &\le 1 + \frac{t^2}{2} + \frac{(\frac{t^2}{2})^2}{2!} + \frac{(\frac{t^2}{2})^3}{3!} + \ldots \\ &\le e^{\frac{t^2}{2}}. \end{aligned}$$

Therefore,

$$E\left[e^{tS_n}\right] = E\left[e^{t\sum X_i}\right] = (E\left[e^{tX_i}\right])^n \le e^{\frac{nt^2}{2}}$$

and,

$$\Pr[|S_n| \ge \lambda] \le \frac{E\left[e^{tS_n}\right]}{e^{t\lambda}} \le \frac{e^{\frac{nt^2}{2}}}{e^{t\lambda}}.$$

Choosing t to be $\frac{\lambda}{n}$, (the optimal choice), the above implies that $\Pr[|S_n| \ge \lambda] \le e^{-\frac{\lambda^2}{2n}}$. □

Another example is the random variable X_i,

$$X_i = \begin{cases} -p & \text{with probability } 1-p \\ 1-p & \text{with probability } p \end{cases}$$

Again, we define $S_n = \sum_{i=1}^n X_i$.

Claim 1.3.2 $\Pr[|S_n| \ge \lambda] \le e^{-\frac{2\lambda^2}{n}}$.

Proof:

$$E\left[e^{tX_i}\right] = (1-p)e^{-tp} + pe^{t(1-p)}$$

Spencer & Alon ([AS1], p. 235) prove that $(1-p)e^{-tp} + pe^{t(1-p)} \le e^{\frac{t^2}{8}}$.
Using this we get that,

$$\Pr[|S_n| \ge \lambda] \le \frac{E\left[e^{tS_n}\right]}{e^{t\lambda}} \le \frac{e^{\frac{nt^2}{8}}}{e^{t\lambda}}.$$

Now, choosing t to be $\frac{4\lambda}{n}$ (the optimal choice), we get

$$\Pr[|S_n| \ge \lambda] \le e^{-\frac{2\lambda^2}{n}}$$

$\square$

1.4 The Inclusion-Exclusion Principle

Let $A_1, A_2, \ldots, A_n$ be events that are not necessarily disjoint. The *Inclusion-Exclusion principle* states that

$$\Pr\left[\bigcup_{i=1}^{n} A_i\right] = \sum_i \Pr[A_i] \;-\; \sum_{i<j} \Pr[A_i \cap A_j] \;+\; \sum_{i<j<k} \Pr[A_i \cap A_j \cap A_k] \;-\; \ldots \;+$$

$$(-1)^{k+1} \sum_{i_1<i_2<\ldots<i_k} \Pr\left[\bigcap_{j=1}^{k} A_{i_j}\right] \;+\; \ldots$$

Bonferroni's inequalities:
For every odd k,

$$\Pr\left[\bigcup_i A_i\right] \le \sum_{j=1}^{k} (-1)^{j+1} \sum_{i_1<\ldots<i_j} \Pr\left[\bigcap_{l=1}^{j} A_{i_l}\right]$$

For every even k,

$$\Pr\left[\bigcup_i A_i\right] \ge \sum_{j=1}^{k} (-1)^{j+1} \sum_{i_1<\ldots<i_j} \Pr\left[\bigcap_{l=1}^{j} A_{i_l}\right]$$

Special cases of these inequalities are :

1. "Markov"-type inequality: $\Pr[\bigcup_i A_i] \le \sum_i \Pr[A_i]$.

2. "Chebyshev"-type inequality: $\Pr[\bigcup_i A_i] \ge \sum_i \Pr[A_i] - \sum_{i<j} \Pr[A_i \cap A_j]$.

1.5 Wald's Equation

Consider the following problem, which comes up in many applications. Let $Z_1, Z_2, \ldots$ be a sequence of random variables, and suppose that we wish to compute $E\left[\sum_{i=1}^{T} Z_i\right]$, where T is a random variable. Wald's Equation provides us with sufficient conditions under which this expectation can be evaluated. We first need to define a *stopping rule.*

Definition 1.5.1 *Suppose $Z_1, Z_2, \ldots$ is a sequence of independent random variables, and let T be a random variable attaining integral values. The variable T is called a stopping rule relative to the sequence $\{Z_i\}$, if the event $T = i$ is independent of $Z_{i+1}, Z_{i+2}, \ldots$*

The idea is that the variables Z_i are observed one at a time: first Z_1, then Z_2, and so on. The variable T represents the number of variables observed when we stop. Obviously, the event $T = i$ must be independent of $Z_{i+1}, Z_{i+2}, \ldots$ Notice that if the sequence $\{Z_i\}$ is not independent, then the event $T = i$ may depend on some variable Z_j, where $j > i$.

Theorem 1.5.1 (Wald's Equation) *Let $Z_1, Z_2, \ldots$ be a sequence of independent random variables with finite expectation μ, and suppose a random variable T with finite expectation is a stopping rule relative to $\{Z_i\}$. Then,*

$$E\left[\sum_{i=1}^{T} Z_i\right] = \mu E[T]$$

Proof: We define the following indicator variables I_i, for $i \geq 1$, as follows:

$$I_i = \begin{cases} 1 & \text{if } i \leq T \\ 0 & \text{otherwise} \end{cases}$$

Clearly,

$$\sum_{i=1}^{T} Z_i = \sum_{i=1}^{\infty} Z_i \cdot I_i$$

and hence,

$$E\left[\sum_{i=1}^{T} Z_i\right] = E\left[\sum_{i=1}^{\infty} Z_i \cdot I_i\right] = \sum_{i=1}^{\infty} E[Z_i \cdot I_i]$$

Where the last equality follows from countable additivity for expectations, since $E[T] < \infty$ implies $\Pr[T = \infty] = 0$, and thus $\sum_{i=1}^{\infty} E\left[Z_i \cdot I_i\right] < \infty$ (see Fact 1.1.1). Since $I_i = 1$, if and only if $i \leq T$, it follows that the value of I_i is completely determined by observing the variables $Z_1, \ldots, Z_{i-1}$. Since Z_i is independent of $Z_1, \ldots, Z_{i-1}$, it follows that I_i and Z_i are independent. Hence,

$$E\left[\sum_{i=1}^{T} Z_i\right] = \sum_{i=1}^{\infty} E[Z_i] \cdot E[I_i] = \mu \sum_{i=1}^{\infty} E[I_i]$$

Notice that $T = \sum_{i=1}^{\infty} I_i$, and thus, by countable additivity for expectations (Fact 1.1.1),

$$E[T] = E\left[\sum_{i=1}^{\infty} I_i\right] = \sum_{i=1}^{\infty} E[I_i]$$

This yields that,

$$E\left[\sum_{i=1}^{T} Z_i\right] = \mu E[T]$$

$\square$

1.6 The Coupon Collector Problem

Suppose we have a box with n different balls, or coupons, labeled $1 \ldots n$. In each iteration we draw a ball, uniformly at random, from the box and then return it. Let T be the random variable that counts the number of draws until we see all n balls. We study the distribution of the random variable T.

We first compute the expected value and variance of T. Let the random variable Y_i indicate the number of balls that we have to draw in order to see a new ball, assuming i balls have already been seen. Clearly $T = Y_0 + Y_1 + \ldots + Y_{n-1}$. By linearity of expectation

$$E[T] = E\left[\sum_{i=0}^{n-1} Y_i\right] = \sum_{i=0}^{n-1} E[Y_i]$$

A non-negative random variable X is said to be geometrically distributed with parameter p if $\Pr[X = k] = p^{k-1} \cdot q$, for all $k \geq 0$. We claim that Y_i is geometrically distributed with parameter $\frac{n-i}{n}$. This follows from the fact that if $Y_i = k$, then in the first $k-1$ draws we did not see a new ball.

$$E[Y_i] = \sum_{k=1}^{\infty} kpq^{k-1} = \frac{p}{(1-q)^2} = \frac{1}{p} = \frac{n}{n-i}$$

$$E[T] = \sum_{i=0}^{n-1} \frac{n}{n-i} = n \sum_{j=1}^{n} \frac{1}{j} = nH_n \approx n \log n$$

Hence, we expect to make about $n \log n$ draws to see all the balls.

The variance of T: if X is geometrically distributed then,

$$\begin{aligned} E\left[X^2\right] &= E\left[X\left(X-1\right)\right] + E\left[X\right] \\ E\left[X\right] &= \frac{1}{p} \\ E\left[X\left(X-1\right)\right] &= \sum_{x=1}^{\infty} x(x-1)pq^{x-1} = pq\sum_{x=2}^{\infty} x(x-1)q^{x-2} \\ &= pq\sum_{x=2}^{\infty}\left(q^x\right)''_p = pq\left(\sum_{x=2}^{\infty} q^x\right)''_p \\ &= pq\left(\frac{q^2}{p}\right)''_p = \frac{2q}{p^2} \end{aligned}$$

Therefore $E\left[X^2\right] = \frac{2q}{p^2} + \frac{1}{p} = \frac{2-p}{p^2}$ and,

$$\mathrm{VAR}\left[X\right] = E\left[X^2\right] - E\left[X\right]^2 = \frac{2-p}{p^2} - \left(\frac{1}{p}\right)^2 = \frac{q}{p^2}.$$

Notice that the random variables Y_i are independent. Therefore,

$$\begin{aligned} \mathrm{VAR}\left[T\right] &= \mathrm{VAR}\left[\sum_i Y_i\right] = \sum \mathrm{VAR}\left[Y_i\right] \\ &= \sum_{i=0}^{n-1} \frac{i}{n}\frac{n^2}{(n-i)^2} = \sum_{i=0}^{n-1}\frac{ni}{(n-i)^2} \\ &= n\sum_{i=0}^{n-1}\frac{i}{(n-i)^2} = n\sum_{j=1}^{n}\frac{n-j}{j^2} \\ &= n^2\sum_{j=1}^{n}\frac{1}{j^2} - n\sum_{j=1}^{n}\frac{1}{j} \approx n^2\cdot\frac{\pi^2}{6} - n\log n. \end{aligned}$$

This means that the standard deviation of T is bn for some small constant $b > 1$. To estimate the probability of deviating from the mean, we substitute in Chebyshev's Inequality,

$$\Pr[|T - nH_n| \geq \lambda bn] \leq \frac{1}{\lambda^2}.$$

Which implies that the probability of being more than $O(\log n)$ standard deviations away from the mean is polylogarithmically small.

We are now ready to calculate the probability distribution of the number of independent draws until all coupons are recorded. We define $p(t) \stackrel{def}{=} \Pr[T \leq t]$. Let the event A_i be defined as follows: coupon i was not recorded in the first t draws. Then,

$$\Pr[A_i] = \left(1 - \frac{1}{n}\right)^t$$

Clearly, $p(t)$ is the probability that none of these event occurs. We can use the Inclusion-Exclusion principle to calculate $p(t)$:

$$p(t) = \Pr[\bigcap_{i=1}^{n} \overline{A_i}] = 1 - \Pr[\bigcup_{i=1}^{n} A_i]$$

$$\Pr[\bigcup_{i=1}^{n} A_i] = \sum_i \Pr\left[Ai\right] \;-\; \sum_{i<j} \Pr\left[A_i \cap A_j\right] \;+\; \sum_{i<j<k} \Pr\left[A_i \cap A_j \cap A_k\right] \;-\; \ldots \;+$$

$$(-1)^{k+1} \sum_{i_1<i_2<\ldots<i_k} \Pr\left[\bigcap_{j=1}^{k} A_{i_j}\right] \;+\; \ldots$$

It is easy to see that the value of $\Pr[\bigcap_{j=1}^{m} A_{i_j}]$ for all m is equal to $(\frac{n-m}{n})^k$. However, calculating a closed formula for $p(t)$ is quite cumbersome. Instead, we approximate $p(t)$ by using the "Poisson approximation method".

Definition 1.6.1 *A random variable X has* Poisson distribution with parameter λ *if*

$$\Pr[X = k] = \frac{\lambda^k e^{-\lambda}}{k!}$$

for $k = 0, 1, 2, \ldots$

Fact 1.6.1 (The Poisson approximation to the Binomial distribution). *For $N \to \infty$ and $N \cdot p$ "not too large", the binomial distribution with parameters N and p can be approximated by the Poisson distribution with parameter $\lambda = N \cdot p$.*

Recall that event A_i happens if coupon i did not show up in the first t draws. Hence, approximating the probability of the event A_i by the Poisson distribution is the same as approximating the event of "all failures" in the binomial distribution with parameters $N = t$ and $p = \frac{1}{n}$. We get:

$$\Pr[A_i] = \binom{N}{0} \left(\frac{1}{n}\right)^0 \left(1 - \frac{1}{n}\right)^t = \left(1 - \frac{1}{n}\right)^t \approx e^{-\frac{t}{n}}$$

The advantage of using the Poisson approximation is that it allows us to regard the events A_i as independent when $n \to \infty$. (Although they are not even pairwise independent!) Formally, we claim:

Claim 1.6.1

$$\lim_{n\to\infty} \Pr[A_j | A_{i_1}, \ldots, A_{i_k}] = \Pr[A_j]$$

Proof:

$$\Pr[A_j|A_{i_1},\ldots,A_{i_k}] = \frac{\Pr[A_j \cap A_{i_1} \cap \ldots \cap A_{i_k}]}{\Pr[A_{i_1} \cap A_{i_2} \cap \ldots \cap A_{i_k}]} = \frac{(1-\frac{k+1}{n})^t}{(1-\frac{k}{n})^t} \approx \frac{e^{-\frac{-t(k+1)}{n}}}{e^{-\frac{tk}{n}}} = e^{-\frac{t}{n}} = \Pr[A_j]$$

$\square$

The probability function we get for T is:

$$\Pr[T \leq t] = \Pr[\text{None of the events } A_i \text{ occurred}] = (1-e^{-\frac{t}{n}})^n \approx e^{-ne^{-\frac{t}{n}}}$$

This probability can be rewritten as the probability of being c standard deviations away from the expectation. We get:

$$\Pr[T \leq n\log n + cn] \approx e^{-ne^{-\frac{n\log n+nc}{n}}} = e^{-e^{-c}}$$

Chapter 2

Linearity of Expectation

The probabilistic method is a useful tool for proving the existence of combinatorial objects having certain properties. This is done by defining a probability space in which the probability that an object sampled at random has the desired property is strictly positive. The existence proofs in this chapter are based on a simple principle, *linearity of expectation*. The expected value of the sum of random variables is equal to the sum of the expected values of the random variables.

2.1 Ramsey number

Definition 2.1.1 *Let K_n denote a clique of size n. Define the Ramsey number $R(k,l)$ to be the smallest n such that every 2-coloring of the edges of K_n with blue and red contains either a red K_k or a blue K_l.*

Ramsey proved that $R(k,l)$ is finite. Here we develop a lower bound for $R(k,k)$. Note that for all $k < l$ $R(k,k) \leq R(l,k) = R(k,l) \leq R(l,l)$.

Theorem 2.1.1

$$R(k,k) \geq 2^{k/2}$$

Proof: We use the probabilistic method for the lower bound as follows: we show that if the graph is small enough, then if we choose a random coloring, the probability of having neither a red K_k nor a blue K_l is positive. Thus, we conclude that for small enough graphs, such a coloring exists, and therefore the Ramsey number cannot be very small.

Let us choose a coloring for the edges of K_n uniformly at random: Pr[Edge e is red] = Pr[Edge e is blue] $= \frac{1}{2}$. For a set of vertices S, such that $|S| = k$, we say that the event A_S occurs if this set is monochromatic. Define an indicator random variable

$$X_S = \begin{cases} 1 & \text{If } A_S \text{ occurs} \\ 0 & \text{Otherwise} \end{cases}$$

For any S we have

$$\Pr[A_S] = \Pr[X_S = 1] = 2(\frac{1}{2})^{\binom{k}{2}} = 2^{1-\binom{k}{2}}$$

The probability that none of the sets S is monochromatic is $\Pr[\bigcap_S \overline{A_S}]$ by definition. For all $n < R(k,k)$, the following must hold:

$$\Pr[\bigcap^{S} \overline{A}_S] = 1 - \Pr[\bigcup_S A_S] > 0$$

which implies that

$$\Pr[\bigcup_S A_S] < 1$$

We approximate $\bigcup_S A_S$ by the sum of the events A_S.

$$\Pr[\bigcup A_S] \le \sum_S \Pr[A_S] = \binom{n}{k} 2^{1-\binom{k}{2}} < 1$$

We want to choose the maximal n satisfying

$$\binom{n}{k} < 2^{\frac{k(k-1)}{2}}$$

Approximating $\binom{n}{k} \approx n^k$ and $2^{\frac{k(k-1)}{2}} \approx \left(\frac{1}{2}\right)^{k^2}$ we get

$$n^k < 2^{k^2/2}$$

$$n < 2^{k/2}$$

Thus, $R(k,k) \ge 2^{\frac{k}{2}}$ □

2.2 Property B

An n-uniform hypergraph has property B if there is a 2-coloring of the vertices such that none of the edges is monochromatic. Let $m(n)$ be the least number of edges of an n-uniform hypergraph that does not possess property B.

Theorem 2.2.1

$$m(n) \geq 2^{n-1}$$

Suppose to the contrary that $m(n) < 2^{n-1}$. Then there is a hypergraph with less than 2^{n-1} edges that does not have property B. Color the vertices randomly and independently by 2 colors with equal probability. Let A_e be the event that edge e is monochromatic. Then, $\Pr[A_e] = 2\left(\frac{1}{2}\right)^n = 2^{1-n}$. Therefore,

$$\Pr[\text{there exists a monochromatic edge}] = \Pr[\bigcup_e A_e] \leq \sum_e \Pr[A_e] = |E|2^{1-n} < 1$$

The last inequality is due to the assumption that $|E| < 2^{n-1}$. Therefore, there exists a coloring such that no edge is monochromatic, meaning that the hypergraph has property B, in contradiction to the assumption. □

2.3 Hamiltonian paths in a tournament

The following observation is immediate.

Fact 2.3.1 *Let X be a positive random variable with more than one value in its support set. There must exist values x_1 and x_2 such that $\Pr[x_1] > 0$, $\Pr[x_2] > 0$ and $x_1 \leq E[X]$, $x_2 \geq E[X]$*

Definition 2.3.1 *A complete directed graph is called a tournament.*

Definition 2.3.2 *A Hamiltonian path in a graph $G(V, E)$ is a path that visits each vertex of V exactly once.*

Proposition 2.3.1 *Every tournament has a Hamiltonian path.*

Proof sketch: By induction on the number of vertices in the tournament. Suppose that in every tournament of size n or less, there is a Hamiltonian path. Given a tournament $T(V, E)$ with $n+1$ vertices, select a vertex v and define two sets: $A \stackrel{def}{=} \{u : (v \to u) \in E\}$ and $B \stackrel{def}{=} \{u : (u \to v) \in E\}$. The subgraphs induced by these sets are two distinct tournaments. By the induction hypothesis each of these graphs contains a Hamiltonian path. Connecting the two paths through v we get a Hamiltonian path in T. □

Theorem 2.3.1 *There exists a tournament of size n with $n! \cdot 2^{n-1}$ Hamiltonian paths.*

Proof: Generate a random tournament T by randomly choosing a direction for the edges of K_n. Let σ be a permutation of the vertices V. Define a random variable X_σ as follows:

$$X_\sigma = \begin{cases} 1 & \sigma \text{ defines a path on G} \\ 0 & \text{otherwise} \end{cases}$$

For all σ, $\Pr[X_\sigma = 1] = \left(\frac{1}{2}\right)^{n-1}$. Let X be a random variable counting the number of Hamiltonian paths in T.

$$X = \sum_{\sigma \in \mathrm{Sym(n)}} X_\sigma$$

We now calculate the expectation of X, and use Fact 2.3.1 to derive the theorem.

$$E[X] = E[\sum X_\sigma] = \sum_\sigma E[X_\sigma] = n! \left(\frac{1}{2}\right)^{n-1}$$

□

2.4 Existence of large cuts

Let $G = (V, E)$ be a graph. A cut in a graph is defined to be the set of edges connecting any two subsets of the vertices. We show that for every graph G, there exists a cut containing at least half of the edges. Color the vertices of G with two colors and remove the edges connecting vertices of the same color. Call the resulting bipartite graph $G' = (V, E')$.

Claim 2.4.1 *There exists a coloring such that* $|E'| \geq \frac{|E|}{2}$.

Proof: Randomly and independently color the vertices with equal probability. For $e \in E$ define

$$X_e = \begin{cases} 0 & e \notin E' \\ 1 & e \in E' \end{cases}$$

Let $X = \sum_{e \in E} X_e$. Clearly $X = |E'|$. Using Fact 2.3.1 it suffices to prove that $E[X] = \frac{|E|}{2}$.

$$E[X] = E\left[\sum_{e \in E} X_e\right] = \sum_e E[X_e] = \sum_{e \in E} \frac{1}{2} = \frac{|E|}{2}$$

□

Notice that for this proof, choosing a coloring which is only pairwise independent suffices. This fact will be useful in Chapter 11 for finding a cut with the above properties.

2.5 Short sum of vectors

Definition 2.5.1 *Given a vector $\vec{V} = (v_1, \ldots, v_n)$ the length of $\vec{V}$ is defined as*

$$\| \vec{V} \| = \sqrt{\sum_{i=1}^{n} v_i} = \sqrt{\vec{V} \cdot \vec{V}}$$

Consider the following problem: we are given n vectors, $\vec{V}_1, \ldots \vec{V}_n$. Each contains n entries, which might be 1 or -1 (so the length of each of them is $\sqrt{n}$). We have to add them, but we may replace each $\vec{V}_i$ with its negation, $-\vec{V}_i$. The following theorem asserts that we can always make these decisions in a way that the sum of these vectors will be relatively short. Notice that if the vectors are mutually orthogonal, ($\vec{V}_i \cdot \vec{V}_j = 0$), then this length is the shortest possible.

Theorem 2.5.1 *Given any n vectors $\vec{V}_1, \ldots \vec{V}_n$ in $\{1, -1\}^n$, there are scalars, $\epsilon_1, \ldots \epsilon_n \in \{-1, 1\}$ such that $\| \sum \epsilon_i \vec{V}_i \| \leq n$*

Proof: The proof will be based on a standard probabilistic argument. We will evaluate the expected length of the sum, while making the decision between $\vec{V}_i$ and $-\vec{V}_i$ at random. It will be shown that the expected square of the length is less than n^2.

Let us choose the ϵ_i-s independently at random to be -1 or 1 with probability 1/2. We evaluate the expected length of $\sum \epsilon_i \vec{V}_i$ by computing the square of that quantity. We use the fact that ϵ_i and ϵ_j are independent and therefore $E[\epsilon_i \epsilon_j] = E[\epsilon_i]\ E[\epsilon_j] = 0$.

$$E\left[\| \sum \epsilon_i \vec{V}_i \|^2\right] = E\left[\sum_{i=1}^{n} \sum_{j=1}^{n} \epsilon_i \epsilon_j \vec{V}_i \cdot \vec{V}_j\right]$$

$$= \sum_{i=1}^{n} E\left[\epsilon_i^2 \vec{V}_i \cdot \vec{V}_i\right] + E\left[\sum_{i=1}^{n} \sum_{\substack{j=1 \\ j \neq i}}^{n} \epsilon_i \epsilon_j \vec{V}_i \cdot \vec{V}_j\right] = \sum_{i=1}^{n} E\left[\vec{V}_i \cdot \vec{V}_i\right] + \sum_{i=1}^{n} \sum_{\substack{j=1 \\ j \neq i}}^{n} \vec{V}_i \cdot \vec{V}_j E[\epsilon_i \epsilon_j]$$

$$= \sum_{i=1}^{n} n + \sum_{i=1}^{n} \sum_{\substack{j=1 \\ j \neq i}}^{n} \vec{V}_i \cdot \vec{V}_j \times 0 = n^2$$

$\square$

Chapter 3

The Deletion Method

Suppose we want to prove that a certain combinatorial object $\mathcal{F}$ exists via the *deletion method.* The proof is conceptually built of two stages: we first show that an object $\mathcal{F}'$ which is very "close" (in a certain sense) to $\mathcal{F}$ exists. Then, we change $\mathcal{F}'$ to obtain $\mathcal{F}$ and prove that the probability of existence (after the change) remains positive. This method is useful in cases where applying a direct probabilistic proof seems difficult.

3.1 The "min-max triangle" Problem

Consider the following problem: Let $S = \{p_1, \ldots p_n\}$ be a set of n points located in a unit square in the plane. Consider all the triangles whose vertices are points of S (there are $\binom{n}{3}$ such triangles). Let $T(S)$ be the area of the triangle(s) with minimum area. The following theorem asserts that for any n there is a set S of n points such that $T(S)$ is not too small.

Theorem 3.1.1

$$\forall n\, \exists S = \{p_1, \ldots p_n\},\ \ \textit{S is contained in the unit square, s.t.}\ \ T(S) \geq \frac{1}{100n^2}.$$

Proof: We prove the theorem by choosing $2n$ points at random in the unit square, testing all the triangles they define, and eliminating the triangles that are "too small" by dropping one vertex from each of them. The core of the proof is the fact that we are left with enough (namely n) points.

Choose $2n$ points in the unit square with uniform distribution. Let p, q, r denote three of them. Define $\mu(p, q, r)$ to be the area of the triangle that p, q and r define. Recall that the area of a triangle is half the length of the base, multiplied by the height of the triangle.

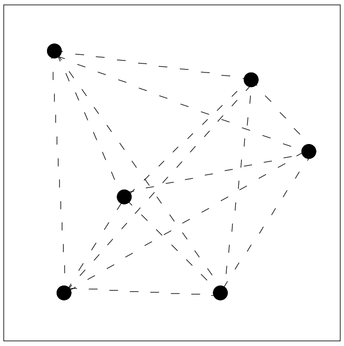


Figure 3.1: 6 points determine 20 triangles

Let $\| p - q \|$ denote the Euclidean distance between p and q.

For any real numbers b and Δb for which $0 \leq b,\ \Delta b \leq 1$ we have

$$\Pr[b \leq \| q - p \| \leq b + \Delta b] \ \leq \ \pi(b+\Delta b)^2 - \pi b^2 \ = \ 2\pi b \Delta b + (\Delta b)^2$$

The inequality follows from the fact that once you fix p, q should fall into a ring defined by two cycles of radii b and $b + \Delta b$ centered at p (see Figure 3.2). Notice that the ring might not be completely contained in the square, and therefore an inequality holds.

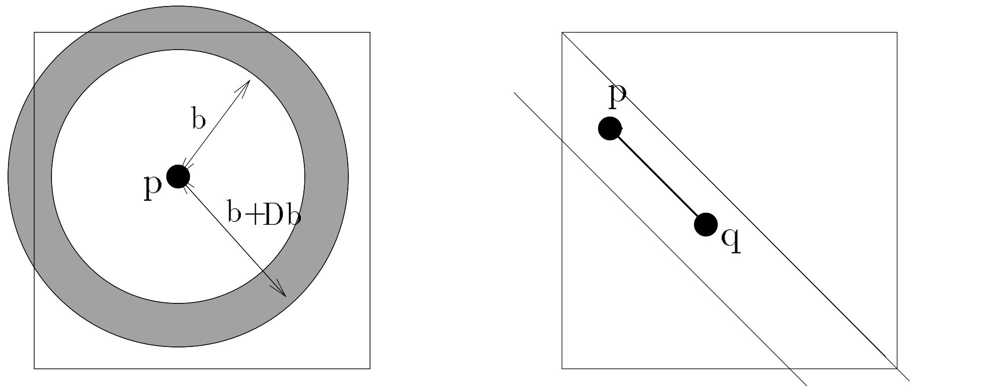
Let (p, q) be the base of the triangle (p, q, r), and let $\| p - q \| = b$. We would like to estimate the probability $\Pr[\mu(p, q, r) \leq \epsilon]$ for some ϵ. This event happens when the height of the triangle h satisfies $h \leq \frac{\epsilon}{2b}$, and so r must not be farther than $\frac{\epsilon}{2b}$ from the line passing through p and q. The probability that this happens is less then $\frac{2\sqrt{2}\epsilon}{b}$, since r must fall in a strip of width $\frac{\epsilon}{b}$ and length less than $\sqrt{2}$.

Therefore,

$$\Pr[\mu(p,q,r) \leq \epsilon] = \int_{b=0}^{\sqrt{2}} \Pr[b \leq \| p - q \| \leq b + db] \ \times \ \Pr[\text{triangle's height} \leq \epsilon/2b] \ =$$

$$\int_{b=0}^{\sqrt{2}} (2\pi b) \frac{4\sqrt{2}\epsilon}{b} db = 16\pi\epsilon$$

Let us now compute the expected number of triangles that have area $\leq \epsilon = \frac{1}{n^2}$. Let S' be a set of $2n$ points uniformly distributed in the unit square. For each triple $p_i, p_j, p_k \in S$, let X_{p_i,p_j,p_k}, be a random variable assuming the value 1 if the area of the triangle defined by (p_i, p_j, p_k) is less than $\epsilon = \frac{1}{n^2}$, and zero otherwise. The probability that the area of some specific triangle is less than $\frac{1}{100n^2}$ is less then $0.6/n^2$. This is also the expected value of



The ring in which q should lie

The strip in which r should lie

Figure 3.2: where points should lie

X_{p_i,p_j,p_k}. Letting X be the number of triangles with area less then $\frac{1}{100n^2}$ we have

$$E[X] \;=\; \sum_{p,q,r\,\in S'} E[X_{p,q,r}] \;\leq\; \binom{2n}{3} 0.6n^{-2} \;\leq n$$

Now, by throwing away an arbitrary vertex from each of these "small area" triangles, we are left with a new set S'' of expected size no less than n, in which no "small area" triangle exists. Therefore, there exists a set S'' of size n for which $T(S'') \geq \frac{1}{100n^2}$ $\square$

3.2 Turan's Theorem

Definition 3.2.1 *An independent set in a graph $G = (V, E)$ is a subset of vertices of V such that no two vertices in the set are adjacent. We denote by $\alpha(G)$ the size of the largest independent set in G.*

The following theorem relates the cardinality of the largest independent set in a graph and the number of edges it contains.

Theorem 3.2.1 (Turan) If $\mid V \mid= n$ and $\mid E \mid= nk/2$ *then* $\alpha(G) \geq \frac{n}{k+1}$.

We prove a weaker result.

Theorem 3.2.2 If $\mid V \mid= n$ and $\mid E \mid= nk/2$ then $\alpha(G) \geq \frac{n}{2k}$.

Proof: The use of probabilistic arguments to prove that a graph contains a large independent set seems a bit bizarre, since everything is fixed, so where can randomness come from? The trick is to choose at random a subset of the vertices that would be a candidate independent set. Using probabilistic arguments, we show that there is a subset that has many more vertices than edges. Then, dropping one vertex from each of these edges, will surely leave us with an independent set. The delicate part is how to find the exact probability for a vertex to be in the subset.

Let us choose a random sample $S \subset V$, where each vertex is chosen independently with probability p (to be specified later). We have

$$E[\,|S|] = np$$

Consider the subgraph G' induced by S. For S to be an independent set, G' cannot contain any edges. For every $e \in E$, define Y_e to be 1 if $e \in E(G')$ and 0 otherwise. Then, $E[Y_e] = p^2$ since an edge belongs to $E(G')$ iff both of its two endpoints are in S which happens with probability p^2. Let $Y = |E(G')|$. Let us evaluate $E(Y)$ (The expected number of edges in this "hopefully" independent subset).

$$E[\,Y\,] = E\left[\sum_{e \in E} Y_e\right] = \sum_{e \in E} E[Y_e] = \frac{nk}{2}p^2$$

If we drop all edges from G' (by dropping a vertex from each of them) we are left with an independent set S^* of expected size $E[|S^*|] \;=\; E[|S| - Y]$, and

$$E[\,|S| - Y\,] \;=\; E[\,|S|\,] - E[Y] = np - \frac{nk}{2}p^2$$

We would like to maximize this expression as a function of p. This is a parabola, which attains its maximum at $p = 1/k$ and then

$$E[|S^*|] \;=\; E[\,|S| - Y\,] = n/2k$$

□

3.3 Erdös's Theorem

Definition 3.3.1 *The chromatic number of a graph $G = (V, E)$ is the minimum number of colors needed to color the vertices of G such that adjacent vertices have different colors. We denote the chromatic number of G by $\chi(G)$.*

Definition 3.3.2 *The girth of a graph G, girth(G), is the length of the shortest cycle.*

The next theorem relates the chromatic number of a graph and its girth. Since coloring a tree requires only two colors, it seem reasonable to assume that a graph with large girth has a small chromatic number, since intuitively, this graph is "locally" a tree.

Erdös's Theorem asserts that for any value of the chromatic number, there are graphs with arbitrarily large girth. Actually, all you have to do in order to find them, is to choose a random graph that is big enough.

Remark 3.3.1 *Notice that given a coloring of the graph, any color class (a set of vertices that has the same color) must be an independent set. A counting argument shows that*

$$\chi(G) \cdot \alpha(G) \quad \geq n$$

Theorem 3.3.1 (Erdös) *For any naturals l, k there exists a graph G, s.t. $\chi(G) \geq k$ and $girth(G) \geq l$.*

Proof: Choose a random graph G_{np}.* The delicate point is the choice of p. If p is too small, the chromatic number would be small, since the largest independent set would be very big. If p is too big, the girth is too small. We first study how the expectation of these quantities is related to p, and then show that deleting not too many vertices, gives us the desired girth without decreasing the chromatic number.

Let θ be a parameter, (to be specified later), $\theta < 1/l$ and let $p = n^{\theta-1}$. We choose a random graph G(V,E) from G_{np}.

Let $(b_1, \ldots, b_c)$ be an (ordered) sequence of vertices of G. The probability that there is a cycle $b_1 - b_2 - \ldots - b_c$ in G is p^c. Choose some subset $\{b_1, \ldots, b_c\} \subset V$; there are $(c-1)!$ ways to form cyclic ordered sequences from these vertices. Using arguments similar to those in the previous result, it is easy to show that the expected number of cycles of length c in the graph will be $\binom{n}{c}\frac{(c-1)!}{2}p^c$, and the expected number X of cycles of length no greater then l is

$$E[X] = \sum_{i=3}^{l} \binom{n}{i} \frac{(i-1)!}{2} p^i = \sum_{i=3}^{l} \frac{n!}{(n-i)!2i} p^i \leq \sum_{i=3}^{l} n^i \frac{1}{2i} (n^{\theta-1})^i \leq \sum_{i=3}^{l} \frac{n^{\theta i}}{2i} \leq \sum_{i=3}^{l} (n^{\theta})^i$$

and if we choose $\theta < \frac{1}{l} - \frac{1}{m \ln n}$

$$E[X] \leq C_1 n$$

*a graph with n vertices where each edge is independently chosen with probability p.

for some constant C_1 that can be made as small as we wish by setting m and n large enough.

Therefore, using Markov's Inequality,

$$\Pr[x \geq \gamma] \leq \frac{E[x]}{\gamma}$$

we have

$$\Pr\left[X \geq \frac{n}{2}\right] \leq \frac{C_1 n}{n/2} = C_2$$

Recall that $\chi(G) \geq n\alpha(G)$. So let us evaluate the probability that $\alpha(G) \geq t$ (i.e. there exists an independent set of size at least t)

The probability that a set $v_1, \ldots, v_c$ of vertices is independent is $(1-p)^{\binom{c}{2}}$. Recall that for any two events, A and B $\Pr[A \vee B] \leq \Pr[A] + \Pr[B]$. Hence,

$$\Pr\left[\alpha(G) \geq t\right] \leq \binom{n}{t}(1-p)^{\binom{t}{2}} < n^t e^{\frac{-p(t-1)}{2}}$$

Since $\forall t \geq 0, (1-p)^t < e^{-pt}$. Set $t = \frac{3 \ln n}{p}$. We have

$$n^t e^{\frac{-p(t-1}{2}} \leq n^{\frac{3\ln n}{p}}(e^{\frac{-3\ln n}{2}})^{\frac{3\ln n}{p}} = \frac{1}{\sqrt{n}} e^{\frac{3 ln n}{p}} \leq \frac{C_3}{\sqrt{n}}$$

for some constant C_3 and n big enough.

We can now take n to be large enough so that both events:

- $\alpha(G) \leq t$.
- The number of cycles of length less than l is less than $n/2$.

will have probability greater than $1/2$.

If two events, A and B, both occur with probability larger than $1/2$, then they intersect. Therefore there exists a graph G for which $\alpha(G) \leq t$ and there are at most $n/2$ cycles of length less than l. Now, by throwing away at most $n/2$ vertices from G we destroy each of these cycles and hence create a new graph G^* that does not contain cycles shorter then l.

Deleting a vertex does not increase $\alpha(G)$. Putting it all together:

$$\chi(G^*) \geq \frac{\mid V(G^*) \mid}{\alpha(G^*)} \geq \frac{\mid V(G) \mid /2}{\alpha(G)} \geq \frac{n/2}{3n^{1-\theta}\ln n} = \frac{n^\theta}{6\ln n}$$

By taking n to be large enough, we can force $\chi(G)$ to be greater than k.

Chapter 4

The Second Moment Method

The *second moment method* is a useful tool for determining the *threshold function* of an event, i.e., a threshold such that below it, the probability of the event tends to 0, and above it, it tends to 1.

Definition 4.0.3 *The family of random graphs G_{np} is defined as the family of all graphs where each edge is independently chosen with probability p.*

It turns out that many interesting graph properties have threshold functions in the family of graphs G_{np}. That is, there exists a value p^* such that if $p < p^*$, the probability that a graph sampled from G_{np} has the property tends to 0, and if $p > p^*$, the probability tends to 1.

Let X be a non negative random variable with $E[X] = \mu$ and $\mathrm{VAR}[X] = \sigma^2$. By *Chebyshev's Inequality:*

$$\Pr[|X - \mu| \geq \lambda\sigma] \leq \frac{1}{\lambda^2}$$

If we set $\lambda = \frac{\mu}{\sigma}$, we get

$$\Pr[X = 0] \leq \Pr[(X = 0) \cup (X \geq 2\mu)] = \Pr[|X - \mu| \geq \mu] \leq \frac{\sigma^2}{\mu^2}$$

Thus, if $\mathrm{VAR}[X] = o(E[X]^2)$, we get $\Pr[X = 0] \longrightarrow 0$.

Let $A_1, \ldots, A_n$ be events where $\Pr[A_i] = p$, and let $X_1, \ldots, X_n$ be their *indicator variables* (i.e., $X_i = 1$ if A_i happens and 0 otherwise). Then, $\forall i: \ \Pr[X_i = 1] = p$.
We are about to find sufficient conditions that ensure that the probability that one of the events A_i occurs tends to 0.

We define $X = \sum_{i=1}^{n} X_n$, and denote by $i \sim j$ if events A_i and A_j are dependent and $i \neq j$.
By definition,

$$\mathrm{VAR}\,[X] = \sum_{i=1}^{n} \mathrm{VAR}\,[X_i] + \sum_{i \neq j} \mathrm{Cov}(X_i, X_j)$$

where,

$$\mathrm{Cov}(X_i, X_j) = E[X_i X_j] - E[X_i]E[X_j] \leq E[X_i X_j] \;=\; \Pr[A_i \cap A_j]$$

For a Bernoulli random variable X_i,

$$\mathrm{VAR}\,[X_i] = p(1-p) \leq p = E[X_i]$$

$$\Longrightarrow \mathrm{VAR}\,[X] \leq \sum_{i=1}^{n} E[X_i] + \sum_{i \sim j} \Pr[A_i \cap A_j]$$

setting $\Delta = \sum_{i \sim j} \Pr[A_i \cap A_j]$

$$\Pr[X = 0] \leq \frac{\mathrm{VAR}\,[X]}{E^2[X]} \leq \frac{\Delta}{E^2[X]} + \frac{1}{E\,[X]}$$

$\Longrightarrow$ If $E\,[X] \longrightarrow \infty$ and $\Delta = o(E[X]^2)$, then $\Pr[X = 0] \longrightarrow 0$
We now assume that events $A_1, \ldots, A_n$ are *symmetric*. (Events $A_1, \ldots, A_n$ are called symmetric if for every $i \neq j$ there exists an automorphism of the underlying probability space that sends event A_i to A_j). Then

$$\Delta = \sum_{i \sim j} \Pr[A_i \cap A_j] = \sum_{i \sim j} \Pr[A_i]\Pr[A_j | A_i] = \sum_{i} \Pr[A_i] \sum_{j \sim i} \Pr[A_j | A_i]$$

setting $\Delta^* = \sum_{j \sim i} \Pr[A_j | A_i]$,

$$\Pr[X = 0] \leq \frac{\Delta^* \cdot E\,[X]}{E[X]^2} + \frac{1}{E\,[X]} = \frac{\Delta^*}{E\,[X]} + \frac{1}{E\,[X]}$$

$\Longrightarrow$ If $E\,[X] \longrightarrow \infty$ and $\Delta^* = o(E\,[X])$, then $\Pr[X = 0] \longrightarrow 0$

4.1 The threshold function for clique

Let G be a graph and let W(G) be the largest clique in G.

Theorem 4.1.1 *The event $W(G) \geq 4$ has a threshold function $p^* = n^{-\frac{2}{3}}$.*

Proof: For every subset of four vertices S let A_S be the event that S induces a clique. Let X_S be the indicator variable of A_S. Clearly, for every S, $\Pr[A_S] = p^6$.
We define $X = \sum_S X_S$. Then, $E[X] = \binom{n}{4} \cdot p^6 \approx \frac{n^4 \cdot p^6}{24}$. By Markov's Inequality:

$$\Pr[X \geq 1] \leq \frac{E[X]}{1} \longrightarrow 0$$

$\Longrightarrow$ If $p \ll n^{-\frac{2}{3}}$ $E[X] = o(1)$. Thus the probability that $W(G) \geq 4 \longrightarrow 0$.

We must show now that for $p \gg n^{-\frac{2}{3}}$, $\Pr[W(G) \geq 4] \longrightarrow 1$.

First note that for $p \gg n^{-\frac{2}{3}}$, $E[X] \longrightarrow \infty$.
Next, we note that two cliques are dependent if only if they intersect in at least two vertices. Let S and T be any two subsets of four vertices.

1. If $|S \cap T| = 2$ then $\Pr[A_T | A_S] = p^5$. Note that for any fixed subset S, there are $O(n^2)$ such subsets T.

2. If $|S \cap T| = 3$ then $\Pr[A_T | A_S] = p^3$. Note that for any fixed subset S, there are $O(n)$ such subsets T.

$\Longrightarrow \Delta^* = \sum_{T \sim S} \Pr[A_T | A_S] \approx np^3 + n^2 p^5$.
$\Longrightarrow$ *If* $p \gg n^{-\frac{2}{3}}$, $E[X] \longrightarrow \infty$, $\Delta^* = o(E[X])$, and therefore $\Pr[W(G) \geq 4] \longrightarrow 1$.

We conclude that $p^* = n^{-\frac{2}{3}}$ is the threshold function of the event $W(G) \geq 4$. $\square$

Chapter 5

Monotone Formulae for Majority

We consider here monotone formulae, i.e., with AND and OR gates. In a formula, each gate has fan-out one.

Let $X = \{x_1, \ldots, x_n\}$ be a set of n = 2m Boolean variables.

Definition 5.0.1 *The majority function is defined as follows:*

$$MAJ(X) = \begin{cases} 1 & \sum_{i=1}^{2m} x_i \geq m \\ 0 & \text{otherwise} \end{cases}$$

Valiant [Va2] considered monotone formulae for computing majority.

Theorem 5.0.2 (Valiant) *There is a monotone formula of size $O(n^{5.3})$ and of depth $O(\log n)$ that calculates MAJ(X).*

Proof: We recursively define a sequence $A_0, A_1, \ldots, A_i, \ldots$ of probability distributions of monotone boolean formulae over the variable set X. The formulae in A_i will all have nonzero probability and size $\leq 2^{2i}$.

We show that for an appropriate i, for every input X, the probability that a randomly chosen formula from A_i will not compute $MAJ(X)$ is less than 2^{-n}. Therefore, with positive probability, there exists a formula in A_i that computes the majority function.

The distribution A_i is defined by the following recursive probabilistic construction of a formula F$\in A_i$:

1. If i = 0 then

$$F = \begin{cases} x_j \ (1 \leq j \leq n) & \text{with probability } \frac{2\alpha}{2m-1} \\ 0 & \text{with probability } 1 - \frac{2\alpha n}{2m-1} \quad \text{(otherwise)} \end{cases}$$

Where $\alpha = \frac{3-\sqrt{5}}{2} \approx 0.38$

2. If $i > 0$ choose G^1, G^2, G^3, G^4 from A_{i-1} independently and let

$$F = (G^1 + G^2)(G^3 + G^4)$$

We define

- X_0 - An assignment to X for which MAJ(X) = 0.
- X_1 - An assignment to X for which MAJ(X) = 1.
- f_i - The probability that $F(X_0) = 1$, where F$\in A_i$.
- h_i - The probability that $F(X_1) = 0$, where F$\in A_i$.

Note that f_i and h_i represent the probability of the event that the formula F disagrees with the majority function.

Clearly,

$$f_0 \leq \frac{(m-1)2\alpha}{2m-1} = \frac{\alpha(2m-1)-\alpha}{2m-1} = \alpha - \Omega(\frac{1}{n})$$
$$h_0 \leq \frac{2m\alpha}{2m-1} + 1 - \frac{2n\alpha}{2m-1} = 1 - \alpha - \Omega(\frac{1}{n})$$

To compute f_i we notice that $\Pr[F(X_0) = 1]$ is the probability that at least one of $\{G^1, G^2\}$ and one of $\{G^3, G^4\}$ equals 1. This leads us to the recursive formula

$$f_i = f_{i-1}^4 + 4f_{i-1}^3(1-f_{i-1}) + 4f_{i-1}^2(1-f_{i-1})^2 = f_{i-1}^4 - 4f_{i-1}^3 + 4f_{i-1}^2.$$

In a similar way we get

$$h_i = -h_{i-1}^4 + 2h_{i-1}^2.$$

To finish the proof we need to show that the probability of error decreases rapidly when increasing the depth of the formulae chosen. When the error is smaller than 2^{-n} we are done since there are only 2^n possible assignments. This will happen when $i = O(\log n)$ and therefore, there exists a formula of depth $O(\log n)$ for computing majority.

The only fixed points of these recurrence relations in the range (0,1) are $f_i = \alpha$ and $h_i = 1-\alpha$. Also, the derivative at α of f_i with respect to f_{i-1} and of h_i with respect to h_{i-1} is 4α in both cases.

We now show that we can reduce f_i to any constant in $O(\log n)$ steps. Assuming f_i is a constant, we note that $f_i < 8f_{i-1}^2$ and $h_i < 2h_{i-1}^2$. Hence, if $f_j = x$, then $f_{j+k} < (8x)^{2^k}/8$. Substituting $x = 2^{-5}$, a further $\log_2 n$ iterations will reduce f_i to 2^{-n-1}. Note that the size of the resulting formula is $O(n^{2(1+\log_\gamma 2)}) = O(n^{5.3})$.

We notice that if $f_{i-1} = \alpha - \epsilon$ then $f_i = \alpha - \epsilon f_i'(\alpha) + \epsilon^2 f_i''(\alpha) - \epsilon^3 f_i'''(\alpha) + \epsilon^4 f_i''''(\alpha)$. Setting $\Delta(\epsilon) = \epsilon f_i''(\alpha) - \epsilon^2 f_i'''(\alpha) + \epsilon^3 f_i''''(\alpha)$, $f_i = \alpha - \epsilon(4\alpha - \Delta(\epsilon))$. Note that $\Delta(0) = 0$. We now choose $\gamma < 4\alpha$, then $f_i = \alpha - \epsilon\gamma - \epsilon(4\alpha - \gamma - \Delta(\epsilon))$.

Finally we notice that $f_i''(\alpha) = 12\alpha^2 - 24\alpha + 8 \approx 0.5836$, $f_i'''(\alpha) = 24\alpha - 24 \approx -14.833$ and $f_i''''(\alpha) = 24$, hence $\Delta(\epsilon)$ is a monotone increasing function for $\epsilon > 0$.

We conclude that there exists an ϵ_0 such that for all $\epsilon < \epsilon_0$: $f_{i-1} = \alpha - \epsilon \Longrightarrow f_i < \alpha - \gamma\epsilon$. Under the same hypotheses: $h_{i-1} = 1 - \alpha - \epsilon \Longrightarrow h_i < 1 - \alpha - \gamma\epsilon$.

From the two above equations, we conclude that for some i for which $\frac{1}{n} \cdot \gamma^i \le \epsilon_0$,

$$f_i < \alpha - \epsilon_0 \text{ and } \; h_i < 1 - \alpha - \epsilon_0$$

(I.e., $i = (\log_2 n + \log_2 \epsilon_0)/\log_2 \gamma$).

In some further O(1) steps, both f_i and h_i can be reduced to any constant we wish, such as 2^{-5}.

It follows from the above discussion that if F is chosen at random from the distribution A_i (for an appropriate i),

$$\Pr\left[\bigcup_X (F(X) \neq MAJ(X))\right] \le \sum_X \Pr[F(X) \neq MAJ(X)] \le \frac{1}{2}$$

$\square$

Open research problem: Construct deterministically a depth $O(\log n)$ monotone formula for computing the majority function. Notice that such a construction exists – the sorting network of Ajtai, Komlos and Szemeredi. However, its size is a polynomial of very high degree.

Chapter 6

The Lovàsz Local Lemma

6.1 The Lemma

Let $A_1, \ldots, A_n$ be events in a probability space. A graph $G = (V, E)$ is called a *dependency graph* of $A_1, \ldots, A_n$ if

- $|V| = n$
- $\forall i, j_1, \ldots, j_k \in V$, vertex i is not adjacent to $j_1, \ldots, j_k$, if event A_i is independent of (any combination of) the events $A_{j_1}, \ldots, A_{j_k}$.

Notice that the dependency graph of a given set of events is not necessarily unique (since the independency of a set of events is a sufficient, but not necessary condition for the corresponding set of vertices to be independent). The following theorem is known as the **Lovàsz Local Lemma.**

Theorem 6.1.1 *Let $A_1, \ldots, A_n$ be events in a given probability space, and let $G = (V, E)$ be their dependency graph. Assume that there exist some fixed numbers p and d such that,*

- *$\forall i \in V$, the degree of i is at most d.*
- *$\forall i \in V$, $\Pr[A_i] \leq p$.*
- *$4pd < 1$ (or, alternatively, $ep(d+1) < 1$).*

Then

$$\Pr\left[\bigcap_{i=1}^{n} \overline{A}_i\right] > 0$$

Remark 6.1.1 *The Lovàsz Local Lemma is a useful tool for proving the existence of a combinatorial object having certain properties. The proof usually proceeds by defining an appropriate probability space, and a set of "bad" events $A_1, \ldots, A_n$ that violate the desired properties (of the combinatorial object). Then, it is necessary to show that the conditions of the Local Lemma are satisfied in order to prove that with positive probability, none of the "bad" events occurs.*

Proof: Let S be a subset of $\{1, \ldots, n\}$. We prove, by induction on the size of S, that

$$\forall i \in V, \ \Pr\left[A_i \,\middle|\, \bigcap_{j \in S, j \neq i} \overline{A}_j \right] \leq 2p \tag{6.1}$$

The induction basis is the empty set for which the claim obviously holds. Suppose that for any set $S \subseteq \{1, \ldots, n\}$ of size strictly less then s, the claim holds. Assume that $i = n$, $S = \{1, \ldots, s\}$ and the neighbors of n in G are $1, \ldots, d$, where $d \leq s$. This can be achieved by adding all the "missing" neighbors to S and renaming the vertices in V accordingly. *

Using Bayes' Rule we get

$$\Pr\left[A_n \,\middle|\, \overline{A}_1, \ldots, \overline{A}_s \right] = \frac{\Pr\left[A_n \cap \overline{A}_1 \cap \ldots \cap \overline{A}_d \,\middle|\, \overline{A}_{d+1}, \ldots, \overline{A}_s \right]}{\Pr\left[\overline{A}_1 \cap \ldots \cap \overline{A}_d \,\middle|\, \overline{A}_{d+1}, \ldots, \overline{A}_s \right]}$$

Now, we get the following upper bound on the numerator:

$$\begin{aligned}
\Pr\left[A_n \cap \overline{A}_1 \cap \ldots \cap \overline{A}_d \,\middle|\, \overline{A}_{d+1}, \ldots \overline{A}_s \right] &\leq \Pr\left[A_n \,\middle|\, \overline{A}_{d+1}, \ldots, \overline{A}_s \right] \\
&= \Pr[A_n] \\
&\leq p
\end{aligned}$$

And, using De Morgan's Rule, the induction hypothesis, and the third assumption of the Local Lemma, we get the following lower bound on the denominator:

$$\begin{aligned}
\Pr\left[\overline{A}_1 \cap \ldots \cap \overline{A}_d \,\middle|\, \overline{A}_{d+1}, \ldots, \overline{A}_s \right] &= 1 - \Pr\left[\bigcup_{i=1}^{d} (A_i \,|\, \overline{A}_{d+1}, \ldots, \overline{A}_s) \right] \\
&\geq 1 - \sum_{i=1}^{d} \Pr\left[A_i \,\middle|\, \overline{A}_{d+1}, \ldots, \overline{A}_s \right] \\
&\geq 1 - 2dp \\
&> 1/2
\end{aligned}$$

*Note that adding vertices to S does not decrease the value of the above expressions.

Therefore

$$\Pr\left[A_n \left| \overline{A}_1, \ldots, \overline{A}_s \right.\right] \leq \frac{p}{1/2} = 2p$$

This completes the proof of Inequality (6.1).

Finally, using Bayes' Rule and Inequality (6.1) we get

$$\Pr\left[\bigcap_{i=1}^{n} \overline{A}_i\right] = \prod_{i=1}^{n} \Pr\left[\overline{A}_i \left| \overline{A}_1 \ldots \overline{A}_{i-1} \right.\right] \geq (1-2p)^n > 0$$

□

6.2 Examples

We now provide two examples for the use of the Lovàsz Local Lemma.

6.2.1 Property B

Recall that a given hypergraph has property B if there exists a 2-coloring of the vertices such that none of the edges is monochromatic.

Claim 6.2.1 *Let H be a hypergraph such that each edge contains at least k vertices and intersects with at most d other edges. Suppose also that $4d \leq 2^{k-1}$. Then, H has property B.*

Proof: Color the vertices uniformly at random by 2 colors. Let A_f be the event that edge f is monochromatic. Then,

$$\Pr[A_f] = 2\,(1/2)^{|f|} \leq (1/2)^{k-1}$$

Denote $p = (1/2)^{k-1}$. Thus, it follows that $\Pr[A_f] \leq p$. By assumption it also follows that $4pd < 1$.

Let G be a dependency graph of the events A_f. Clearly, events that correspond to edges that do not intersect are independent. Hence, the degree of each vertex in G is at most d.

Since the three conditions of the Lovàsz Local Lemma are met, we get

$$\Pr\left[\bigcap_{f} \overline{A}_f\right] > 0$$

Hence, H has property B. □

Claim 6.2.2 *Let H be a k-uniform hypergraph (i.e., each edge contains exactly k vertices) and k-regular (i.e., each vertex participates in exactly k edges). Then, $\forall k \geq 9$, H has property B.*

Proof: Color the vertices uniformly at random by 2 colors. Let A_f be the event that edge f is monochromatic. Since H is k-uniform, we get

$$\Pr[A_f] = 2\,(1/2)^{|f|} = (1/2)^{k-1}$$

Denote $p = (1/2)^{k-1}$. It follows that $\Pr[A_f] \leq p$.

Since H is k-regular, each edge intersects at most $k(k-1)$ other edges. Denote $d = k(k-1)$. Thus, $\forall k \geq 9$, $ep(d+1) = e\,(1/2)^{k-1}\,(k(k-1)+1) < 1$.

Let G be a dependency graph of the events A_f. Clearly, events that correspond to edges that do not intersect are independent. Hence, the degree of each vertex in G is at most d.

Since the three conditions of the Lovàsz Local Lemma are met, we get

$$\Pr\left[\bigcap_f \overline{A}_f\right] > 0$$

Hence, H has property. B $\square$

6.2.2 Ramsey number

Let K_n denote a clique of size n. Recall that $R(k,k)$ is the smallest n such that every 2-coloring of the edges of K_n induces a monochromatic clique K_k.

In a previous section we used the basic probabilistic method to prove that $R(k,k) \geq 2^{\frac{k}{2}}$. In this section, we show how to use the Lovàsz Local Lemma to obtain a similar result.

Let us choose a uniform random 2-coloring of the edges of K_n. $\forall S \subset V$, $|S| = k$, define A_S to be the event that the k-clique induced by S is monochromatic. Then $\Pr[A_S] = 2 \cdot (1/2)^{\binom{k}{2}}$. Denote $p = 2 \cdot (1/2)^{\binom{k}{2}}$.

Let G be a dependency graph of the events A_S. The event A_S depends only on events $A_{S'}$ for which S and S' contain at least one common edge. Hence, the degree of each vertex in G is at most

$$d \leq \binom{k}{2} \cdot \binom{n-2}{k-2}$$

By the Lovàsz Local Lemma, if $4pd < 1$, then there is a positive probability that no subclique of size k is monochromatic in the coloring. Therefore, to get a lower bound on Ramsey number $R(k,k)$, we need the maximal n satisfying

$$4pd = 4 \cdot \binom{k}{2} \cdot \binom{n-2}{k-2} \cdot 2 \cdot (1/2)^{\binom{k}{2}} < 1$$

Thus, we get[†]

$$R(k,k) > n \approx \frac{k}{e}\sqrt{2}\, 2^{k/2}(1 + O(1))$$

6.3 Beck's Algorithm for Property B

Probabilistic proofs that use the Lovàsz Local Lemma also imply a randomized algorithm for constructing the combinatorial object whose existence is proved. However, the probability of success of such an algorithm might be exponentially small, and for many years, an intriguing open problem was to construct a randomized algorithm which has a high probability of success. This problem was solved in the affirmative in 1991 by J. Beck [Bec], (see also [Al2]), who gave a randomized algorithm that finds a coloring of a hypergraph with property B, where the probability of success is polynomially small. We note that this algorithm can be made deterministic using the techniques described in Part III of these lecture notes.

6.3.1 The algorithm

Let H be a hypergraph containing N edges $e_1, \ldots, e_N$ such that $\forall i$, edge e_i intersects at most d other edges and $|e_i| = n$. Denote $D = d(d-1)^3$. Choose a partition $n = n_1 + n_2 + n_3$ such that

- $16D(d+1) < 2^{n_1}$
- $16D(d+1) < 2^{n_2}$
- $2e(d+1) < 2^{n_3}$

(For example, when $d < 2^{n/11}$ the partition $n_1 = n_2 \approx 5n/11, n_3 \approx n/11$ satisfies the above conditions). The following algorithm colors the vertices of H such that no edge is monochromatic (i.e., H has property B). Its expected running time is $O(N(\ln N)^C)$:

[†]This requires some work.

Phase 1 Insert the vertices of H into a queue in some arbitrary order.

1. Repeat until the queue is empty:
   (a) Let v is the head of the queue.
   (b) Color v by red or blue with equal probability.
   (c) If $\exists i$ such that edge e_i contains n_1 vertices of the same color and $n - n_1$ vertices which have not yet been colored:
      - Mark edge e_i as "dangerous".
      - Remove from the queue all the vertices of e_i that have not yet been colored.
2. Delete all edges that contain at least one red vertex and one blue vertex.

Phase 2 Execute Phase 1 with parameter n_2 for each connected component in the resulting hypergraph (i.e., the hypergraph that contains only the vertices that were not yet colored and the edges that were not yet deleted).

Phase 3 Try all possible colorings for the remaining hypergraph. Stop whenever a coloring with property B is found.

6.3.2 Analysis

For all $1 \leq i \leq N$, let A_i be the event that edge e_i is monochromatic. Let G be a dependency graph of the events A_i. Denote by S_1 and S_2 the sets of edges in H that "survive" the first and second phase, respectively. Let $G_{|S_1}, G_{|S_2}$ be the dependency graphs of the "surviving events" from S_1 and S_2, respectively.

Lemma 6.3.1 *The size of each connected component of $G_{|S_1}$ is $O(\ln N)$ with probability tending to 1.*

Proof: Let T be a subset of vertices from the dependency graph G. T is called a 4-*tree* if:

- $\forall i, j \in T$, the distance in G between i and j is at least 4. (In particular, T is an independent set of vertices from G.)
- Let E_T be the following set of edges: $(i, j) \in E_T$ if $i, j \in T$ and the distance in G between i and j is exactly 4. Then, $G_T = (T, E_T)$ is a connected graph.

The probability that an edge e_i of H becomes dangerous in the first phase is 2^{1-n_1}. The probability that an edge e_i of H survives the first phase is bounded by the probability that e_i, or at least one of its neighboring edges, becomes dangerous. Let T be a 4-tree of G, then

$$\begin{aligned}
&\Pr[\text{T survives the first phase}] = \\
&\Pr[\textstyle\bigcap_{i\in T} e_i \text{ survives the first phase}] \leq \\
&\Pr[\textstyle\bigcap_{i\in T} e_i \text{ or one of its neighboring edges becomes dangerous}] = \\
&\textstyle\prod_{i\in T} \Pr[e_i \text{ or one of its neighboring edges becmes dangerous}] \leq \\
&((d+1)(2^{1-n_1}))^{|T|}
\end{aligned}$$

The last equality is due to the fact that T is a 4-tree. Hence, $\forall i, j \in T$, the edges e_i and e_j do not have a common neighboring edge. Thus we get

$$\Pr[\text{T survives the first phase}] \leq \left((d+1)(2^{1-n_1})\right)^{|T|} \tag{6.2}$$

By the definition of a 4-tree, the graph $G_T = (T, E_T)$ is connected, and thus it must contain a spanning tree. By Cayley's Theorem, there can be at most $\frac{|T|^{|T|-2}}{|T|!} < 4^{|T|}$ possible spanning trees on $|T|$ vertices, which are not isomorphic. Also, for a specific spanning tree on $|T|$ vertices, the number of 4-trees containing this tree is bounded by

$$N\left(d(d-1)^3\right)^{|T|-1} = N \cdot D^{|T|-1} < N \cdot D^{|T|}$$

Hence,

$$\text{The number of possible 4-trees on } |T| \text{ vertices} < N(4D)^{|T|} \tag{6.3}$$

Therefore, from (6.2) and (6.3) it follows that the expected number of 4-trees that survive the first phase is

$$N\left(8D(d+1)2^{-n_1}\right)^{|T|} \leq N(1/2)^{|T|}$$

For $|T| \geq C_1 \ln N$ we get that the expected number of 4-trees of size $|T|$ that survive the first phase is $o(1)$ (i.e., tending to 0). Hence, the size of the maximal 4-tree contained in a specific connected component of $G_{|S_1}$ is at most $C_1 \ln N$ with probability tending to 1. This yields that the size of the component itself is at most $d^3 C_1 \ln N = C_2 \ln N = O(\ln N)$ with probability tending to 1. □

Lemma 6.3.2 *The size of each connected component of* $G_{|S_2}$ *is* $O(\ln\ln N)$ *with probability converging to 1.*

Proof: The proof is essentially the same as the proof of Lemma 6.3.1. □

Lemma 6.3.3 *The expected running time of the algorithm is $O(N(\ln N)^C)$.*

Proof: In phases 1 and 2 the vertices are considered one by one and the time required for each vertex is bounded by a constant. Hence, phases 1 and 2 take $O(N)$ time. In phase 3 all possible coloring of each connected component are considered. By Lemma 6.3.2, the expected number of possibilities to color a certain connected component is

$$2^{n \ln \ln N} = (\ln N)^n = O((\ln N)^C)$$

and since there can be at most N connected components, the expected running time of the third phase is $O(N(\ln N)^C)$. □

Lemma 6.3.4 *The algorithm succeeds in producing a coloring with property B with probability tending to 1.*

Proof: The probability of success of phases 1 and 2 can be made to be tending to 1. Hence, it is enough to show that in the third phase there always exists a coloring in which no edge is monochromatic. After executing phases 1 and 2, each of the "surviving" edges of H contains at least $n_3 = n - n_1 - n_2$ vertices that have not yet been colored. Therefore, the probability of the events represented by the vertices of $G_{|S_2}$ is at most 2^{1-n_3}. Denote $p = 2^{1-n_3}$. The degree of vertices in $G_{|S_2}$ is still bounded by d. Recall that the partition $n = n_1 + n_2 + n_3$ was chosen such that $2e(d+1) < 2^{n_3}$. Thus $ep(d+1) < 1$. Hence, the claim follows from the Lovàsz Local Lemma. □

Chapter 7

Martingales

7.1 Definitions

We begin by recalling the notion of conditional expectation.

Definition 7.1.1 *Given random variables X and Y, their conditional expectation, $E[X|Y]$, is defined to be*

$$E[X|Y=y_i] \triangleq \sum_j x_j \cdot \Pr[X=x_j|Y=y_i]$$

Notice that $E[X|Y]$ is a random variable. For any two random variables, X and Y, the properties listed below follow directly from the definition of conditional expectation:

- $E[E[X|Y]] = E[X]$
- $E[X|X=x_i] = x_i$
- $E[X+Y|Z] = E[X|Z] + E[Y|Z]$
- $E[Y \cdot E[X|Y]] = E[X \cdot Y]$
- $E[f(Y) \cdot E[f(X)|Y]] = E[f(X) \cdot f(Y)]$

Definition 7.1.2 *A sequence of random variables $X_0, X_1, \ldots, X_m$ is called a* martingale *if for all $0 \le i \le m-1$*

$$E[X_{i+1}|X_i] = X_i$$

Example: Consider a casino gambler having ∞-credit. Let X_0 be the gambler's initial capital and let X_i be his capital after i bets. Assume that all games are fair (i.e., the expected gain in a bet is 0). Clearly, $E[X_{i+1}|X_i] = X_i$. Hence the sequence $X_0, X_1, \ldots, X_m$ is a martingale.

7.2 Azuma's Inequality

Azuma's Inequality states that under certain assumptions, a martingale has a small probability of drastically changing.

Theorem 7.2.1 (Azuma's Inequality) *Let $X_0, X_1, \ldots, X_m$ be a martingale such that*

- $X_0 = 0$
- $|X_{i+1} - X_i| \leq 1 \quad 0 \leq i \leq m-1$

Then $\forall\ \lambda > 0$

$$\Pr\left[X_m > \lambda\sqrt{m}\right] < e^{-\lambda^2/2}$$

Proof: First we prove the following technical claim that for $Y \in [-1, 1]$

$$e^{\alpha Y} \leq \frac{e^{\alpha} + e^{-\alpha}}{2} + \frac{e^{\alpha} - e^{-\alpha}}{2} Y$$

Consider the following functions of Y:

- $f(Y) = e^{\alpha Y}$
- $g(Y) = \frac{e^{\alpha}+e^{-\alpha}}{2} + \frac{e^{\alpha}-e^{-\alpha}}{2} Y$

Notice that $g(Y)$ is the straight line that passes through the points $(-1, e^{-\alpha})$ and $(1, e^{\alpha})$ and that $f(Y)$ is the exponential function that passes through the same points $(-1, e^{-\alpha})$ and $(1, e^{\alpha})$. Since f is a convex function, it follows that $f(Y) \leq g(Y)$ for $Y \in [-1, 1]$.

Now, for all $1 \leq i \leq m$ let us define $Y_i = X_i - X_{i-1}$. By definition it follows that $|Y_i| \leq 1$ and $E[Y_i|X_{i-1}] = 0$. Hence, using these facts and the above claim we get

$$\begin{aligned} E\left[e^{\alpha Y_i}|X_{i-1}\right] &\leq E\left[\frac{e^{\alpha}+e^{-\alpha}}{2} + \frac{e^{\alpha}-e^{-\alpha}}{2} Y_i \middle| X_{i-1}\right] \\ &= \frac{e^{\alpha}+e^{-\alpha}}{2} \\ &\leq e^{\alpha^2/2} \qquad (7.1) \end{aligned}$$

Notice that $X_m = \sum_{i=1}^{m} Y_i$ and $X_{m-1} = \sum_{i=1}^{m-1} Y_i$. Hence, using (7.1) we get

$$\begin{aligned}
E\left[e^{\alpha X_m}\right] &= E\left[\prod_{i=1}^{m} e^{\alpha Y_i}\right] \\
&= E\left[\left(\prod_{i=1}^{m-1} e^{\alpha Y_i}\right) e^{\alpha Y_m}\right] \\
&= E\left[e^{\alpha X_{m-1}} e^{\alpha Y_m}\right] \\
&= E\left[e^{\alpha X_{m-1}} E\left[e^{\alpha Y_m} | X_{m-1}\right]\right] \\
&\leq E\left[e^{\alpha X_{m-1}}\right] e^{\alpha^2/2}
\end{aligned}$$

By similarly repeating for m times, we get

$$E\left[e^{\alpha X_m}\right] \leq e^{\alpha^2 m/2} \tag{7.2}$$

Hence, by Markov Inequality and (7.2),

$$\Pr\left[X_m > \lambda\sqrt{m}\right] = \Pr\left[e^{\alpha X_m} > e^{\alpha\lambda\sqrt{m}}\right] \leq \frac{E\left[e^{\alpha X_m}\right]}{e^{\alpha\lambda\sqrt{m}}} \leq \frac{e^{\alpha^2 m/2}}{e^{\alpha\lambda\sqrt{m}}}$$

Choosing $\alpha = \lambda/\sqrt{m} > 0$, we get

$$\Pr\left[X_m > \lambda\sqrt{m}\right] \leq e^{-\lambda^2/2}$$

$\square$

The following theorem is a generalization of Azuma's Inequality:

Theorem 7.2.2 *Let $X_0, X_1, \ldots, X_m$ be a martingale such that*

- $X_0 > 0$.
- $|X_{i+1} - X_i| < C_i \quad 0 \leq i \leq m-1$

Then, $\forall\ \lambda > 0$

$$\Pr\left[|X_m - X_0| \geq \lambda\right] \leq e^{\left(\frac{-\lambda^2/2}{\sum_{i=0}^{m-1} C_i^2}\right)}$$

The proof of this theorem is omitted.

7.3 Examples

7.3.1 The chromatic number of random graphs

Recall the following ensemble of distributions of random graphs, denoted by G_{np}. For fixed n and p, the distribution G_{np} includes graphs over n vertices for which each edge is chosen with probability p. In this section we use martingales to bound the variance of the chromatic number of G, $\chi(G)$, where G is drawn from G_{np}.

For a graph H, denote by H_i the induced subgraph of H on the first i vertices. Let G be a random graph that agrees with H_i on the first i vertices, and where the rest of the edges are chosen with probability p. Define the following sequence of functions with respect to a graph H.

$$\chi_i = E\left[\chi(G)\,\middle|G_{[i]} = H_{[i]}\right]$$

In other words, χ_i is the expectation of the chromatic number of the graphs in G_{np}, for which the induced subgraph over the first i vertices is fixed.

Theorem 7.3.1 *The sequence $\chi_1, \ldots, \chi_n$ forms a martingale.*

Proof: We have to show that $E[\chi_{i+1}\,|\chi_i] = \chi_i$. Consider the following n-stage process that unravels graph H: at stage $i+1$, the $i+1$st vertex and all potential edges connecting it to the first i vertices are exposed. Notice that the induced subgraph $H_{[i]}$ is already known at the beginning of stage $i+1$ (in which $H_{[i+1]}$ is exposed).

For ease of exposition, we consider stage $i+1$ of the graph-exposure process as a collection of i "subprocesses" in which the corresponding i candidate edges are exposed. Now let us consider the first candidate edge that is exposed in stage $i+1$. Denote this edge by e. Notice that the probability space of the random variable χ_i contains all possible graphs over i vertices. Let a' be the expectation of the chromatic number of graphs in G_{np} when: (i) $H_{[i]}$ is known; (ii) e exists. Let a'' be the expectation of the chromatic number of graphs in G_{np} when: (i) $H_{[i]}$ is known; (ii) e does not exist. Then,

$$\chi_i = a' \cdot \Pr[e \in H] + a'' \cdot \Pr[e \notin H] = a'p + a''(1-p)$$

But, this is exactly the expectation of the chromatic number of graphs in G_{np} where H_i and e are known. (An example is shown in Fig 7.1.) Using the same argument for all i candidate edges of stage $i+1$, we get that the expectation of the chromatic number of graphs in G_{np} at the end of this stage is also χ_i. Therefore, we get $E[\chi_{i+1}\,|\chi_i] = \chi_i$ $\square$

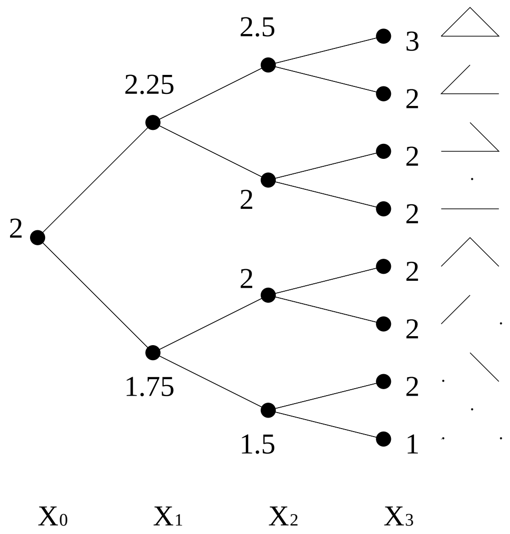


Figure 7.1: The edge exposure process with $n = 3$ where the edges are exposed in the order "bottom,left,right". The values $\chi_i(H)$ are given by tracing from the central node to the leaf labeled H.

The following is due to Shamir and Spencer [SS].

Theorem 7.3.2 *Given H, a random graph in G_{np}, define $C \triangleq E[\chi(G) : G \in G_{np}]$. Then*

$$\Pr\left[|\chi(H) - C| > \lambda\sqrt{n}\right] < e^{-\lambda^2/2}$$

Proof: By definition $\chi_1 = E[\chi(G)]$. By Theorem 7.3.1, the sequence $\chi_1, \ldots, \chi_n$ is a martingale. Also, it is easy to see that $|\chi_{i+1} - \chi_i| \leq 1$ since the addition of a single vertex to H_i increases the chromatic number by at most 1. Hence, using Azuma's Inequality (Theorem 7.2.1), the result follows. $\square$

Corollary 7.3.1 *If λ is chosen to be a function $\omega(n) \stackrel{n\to\infty}{\longrightarrow} \infty$, then the chromatic number of a graph in G_{np} is within an interval of size $2\sqrt{n} \cdot \omega(n)$ around its expectation, with probability tending to 1.*

7.3.2 Dropping balls

Assume we have m balls and n boxes. At each step we take a ball and drop it uniformly at random to one of the boxes. Our aim is to show that a random variable Z, that counts the number of empty boxes, (after all balls are dropped), is highly concentrated around its expectation. The proof presented here follows [MR].

Define a martingale $Z_0, Z_1, \ldots, Z_m$. Let $Z_0 = E[Z]$, and for every $1 \leq t \leq m$, let the random variable Z_t be the expectation of the number of empty boxes, assuming that: (i) we know where the first t balls fell; (ii) dropping the remaining $(m-t)$ boxes uniformly at random. Obviously, $Z_m = Z$.

Claim 7.3.1 *The sequence $Z_0, Z_1, \ldots, Z_m$ is a martingale.*

Proof: Similar to the proof of Theorem 7.3.1. □

Let us compute $E[Z]$. For every box, the probability of being empty after m steps, is $(1-1/n)^m$. Due to linearity of expectation:

$$E[Z] = n\left(1 - \frac{1}{n}\right)^m = Z_0 \ .$$

Denote $\mu = n(1-1/n)^m$.

Theorem 7.3.3 *For every $\lambda \geq 0$ it holds:*

$$\Pr[|Z - \mu| \geq \lambda] < 2e^{\frac{-\lambda^2 n}{n^2 - \mu^2}} \ ,$$

where Z and μ are as defined above.

Proof: We prove this using the generalized Azuma Inequality. To do so, we have to estimate $|Z_{t+1} - Z_t|$, for every $0 \leq t \leq m-1$. Denote by $Z(Y,t)$ the expectation of Z conditioned on the fact that there are Y empty boxes after t steps. Then, obviously,

$$Z(Y,t) = Y_{t-1}\left(1 - \frac{1}{n}\right)^{m-t+1}$$

Define a random variable Y_t, $0 \leq t \leq m$, counting the number of empty boxes after t steps. Then

$$Z_{t-1} = Z(Y_{t-1}, t-1) = Y_{t-1}\left(1 - \frac{1}{n}\right)^{m-t+1}$$

Suppose we have performed $(t-1)$ steps. There are two possibilities.

Case 1: At step t the ball falls into a non-empty box with probability $1 - Y_{t-1}/n$. In this case,

$$\begin{aligned} Y_t &= Y_{t-1} \ , \\ Z_t &= Y_{t-1}\left(1 - \frac{1}{n}\right)^{m-t} \ . \end{aligned}$$

Case 2: At step t the ball falls into an empty box with probability Y_{t-1}/n. In this case,

$$Z_t = (Y_{t-1} - 1)\left(1 - \frac{1}{n}\right)^{m-t}$$

Define $\Delta_t = Z_t - Z_{t-1}$. Then in Case 1:

$$\Delta_t = Y_{t-1}\left(1 - \frac{1}{n}\right)^{m-t} - Y_{t-1}\left(1 - \frac{1}{n}\right)^{m-t+1} = \frac{Y_{t-1}}{n}\left(1 - \frac{1}{n}\right)^{m-t}$$

whereas in Case 2:

$$\begin{aligned}\Delta_t &= (Y_{t-1} - 1)\left(1 - \frac{1}{n}\right)^{m-t} - Y_{t-1}\left(1 - \frac{1}{n}\right)^{m-t+1} \\ &= -\left(1 - \frac{Y_{t-1}}{n}\right)\left(1 - \frac{1}{n}\right)^{m-t}\end{aligned}$$

Since $0 \le Y_{t-1} \le n$ for every t, we have:

$$-\left(1 - \frac{1}{n}\right)^{m-t} \le \Delta_t \le \left(1 - \frac{1}{n}\right)^{m-t}$$

(The lower bound is obtained by taking $Y_{t-1} = 0$; the upper bound is obtained by taking $Y_{t-1} = n$).

Denote $c_t = (1 - 1/n)^{m-t}$, then $|Z_t - Z_{t-1}| \le c_t$ for every $1 \le t \le m$. Now,

$$\sum_{t=1}^{m} c_t^2 = \frac{1 - \left(1 - \frac{1}{n}\right)^{2m}}{1 - \left(1 - \frac{1}{n}\right)^2} = \frac{n^2 - \mu^2}{2n + 1} < \frac{n^2 - \mu^2}{2n} \ ,$$

And so the generalized Azuma Inequality yields:

$$Pr[|Z - \mu| \ge \lambda] < 2e^{-2\lambda^2 / \sum_{t=1}^{m} c_t^2} < 2e^{\frac{-\lambda^2 n}{n^2 - \mu^2}}$$

$\square$

Chapter 8

Expanders

Expanders are graphs that have the property that every subset (which is not too large) is adjacent to many vertices. This property is called the *expansion* property. Such graphs have found numerous applications, in virtually every area of theoretical computer science. The challenge is to generate expanders that contain only a linear number of edges. (The complete graph is clearly an expander). Using the probabilistic method, it is quite easy to show that an expander containing a linear number of edges indeed exists. The first explicit construction of an expander was given by Gaber and Galil [GG], and the best construction to date is by Lubotzky, Phillips and Sarnak [LPS], and [Ma]. We note that all known constructions of expanders use a characterization of the expansion property via the separation between the first and second eigenvalues of the adjacency matrix of the graph. All known constructions of expanders are not hard to describe; however, the proofs that these constructions indeed yield the expansion property, is far beyond the scope of this course, in particular the proof of [LPS]. It is an outstanding open question to find an expander construction that has a "simple" proof.
Notes prepared by Ron Roth were very helpful for the material presented in this chapter.

8.1 Background from Linear Algebra

This section summarizes material from Linear Algebra that will be used in subsequent sections. For more details the reader is referred to any standard textbook on Linear Algebra, and to the book by Biggs [Bi].

Let $\vec{a}$ be a vector in $\Re^n$ (we let $\Re$ denote the real numbers, and an arrow over a variable to denote it is is a vector). The ith coordinate of $\vec{a}$ is denoted by a_i. The sum $\sum_{i=1}^{n} |a_i|$ is denoted by $|\vec{a}|$, and $||\vec{a}||$ denotes $\sqrt{a_1^2 + \cdots + a_n^2}$.

Also, we refer to $||\vec{a}||$ as the *norm* of $\vec{a}$. Multiplication of vectors, matrices, a matrix by a vector (or vice versa), is denoted by juxtaposing the variables denoting the multiplicands. We denote the inner product of $\vec{a}$ and $\vec{b}$ by $\langle \vec{a}, \vec{b} \rangle$.

Let $G = (V, E)$ be an undirected Δ-regular graph on n vertices (a Δ-*regular* graph is a graph in which the degree of each vertex is Δ). Let A be the adjacency matrix of G. That is

$$A_{ij} = \begin{cases} 1 & (i,j) \in E \\ 0 & \text{otherwise.} \end{cases}$$

Obviously, A is symmetric, i.e., $A = A^T$, where A^T denotes the transpose of A. A real number λ is called an *eigenvalue* of A, if there exists a non-zero vector $\vec{x}$ for which $A\vec{x} = \lambda\vec{x}$. The vector $\vec{x}$ is called an *eigenvector* corresponding to λ. The eigenvalues of A are the solutions of the *characteristic equation* $(\lambda I - A)\vec{x} = 0$, i.e., all the λ-s for which $\det(\lambda I - A) \neq 0$. The *algebraic multiplicity* of an eigenvalue is its multiplicity in the solution of the characteristic equation. The *geometric multiplicity* of an eigenvalue is the dimension of the space of eigenvectors corresponding to it. The geometric multiplicity of an eigenvalue is always smaller than or equal to its algebraic multiplicity.

Theorem 8.1.1 *The following holds for every symmetric matrix A over $\Re$.*

1. *All the eigenvalues of A are real.*

2. *The eigenvectors corresponding to different eigenvalues are independent.*

Proof: Suppose A may also contain entries that are complex numbers. The *hermitian* of a matrix A, denoted by A^H, satisfies $A_{ij}^H = \overline{A}_{ji}$ (where $\overline{A}_{ji}$ denotes the conjugate complex of A_{ji}). We prove the first part of the theorem for every hermitian matrix A, i.e., for which $A = A^H$. Obviously, a symmetric matrix (over the real numbers) is hermitian. We need the following claim.

Claim: If $A = A^H$, then for any vector $\vec{x}$ (over the complex numbers), $\vec{x}^H A\vec{x}$ is real.

Proof: Consider the following equalities.

$$(\vec{x}^H A\vec{x})^H = \vec{x}^H A^H \vec{x}^{HH} = \vec{x}^H A^H \vec{x} = \vec{x}^H A\vec{x}.$$

The first equality follows from the fact that $(XY)^H = Y^H X^H$, for any matrices X and Y. The last equality follows from the fact that $A = A^H$. We get that a number c (where $c = \vec{x}^H A\vec{x}$) satisfies $c = c^H$. Therefore c must be real.

(**Proof of Theorem 8.1.1, continued:**) Suppose now that λ is an eigenvalue of A, that is $A\vec{x} = \lambda\vec{x}$, for some vector $\vec{x}$. We have the following equalities.

$$\vec{x}^H A\vec{x} = \vec{x}^H \lambda\vec{x} = \lambda\vec{x}^H\vec{x} = \lambda||\vec{x}||^2.$$

The first equality follows from the fact that λ is an eigenvalue of A. The last equality follows from the definition of a norm of a vector over the complex numbers. The norm of a vector is a real number, so in the above equalities, we have a real number at the left handside, and on the right handside we have λ multiplied by a real number. We conclude that λ is a real number. Now suppose that $A\vec{x} = \lambda_1\vec{x}$ and $A\vec{y} = \lambda_2\vec{y}$, where $\lambda_1 \neq \lambda_2$. There are the following equalities:

$$\lambda_1\vec{x}^T\vec{y} = (\lambda_1\vec{x})^T\vec{y} = (A\vec{x})^T\vec{y} = \vec{x}^T A\vec{y} = \vec{x}^T(\lambda_2\vec{y}) = \lambda_2\vec{x}^T\vec{y}.$$

Since $\lambda_1 \neq \lambda_2$, we must have that $\vec{x}^T\vec{y} = 0$. We conclude that $\vec{x}$ and y are orthogonal and therefore independent. $\square$

Remark 8.1.1 *A symmetric matrix A can be diagonalized by finding a basis of orthogonal eigenvectors. Suppose that these eigenvectors induce the columns of matrix S. Then, $S^{-1}AS$ is a diagonal matrix, where all the eigenvalues appear on the diagonal. In general, a matrix can be diagonalized if and only if the algebraic multiplicity of each eigenvalue is equal to its geometric multiplicity. Hence, for a symmetric matrix A, the algebraic multiplicity of each eigenvalue is equal to its geometric multiplicity.*

Theorem 8.1.2 *Let A be the adjacency matrix of a Δ-regular graph G.*

(i): *Δ is an eigenvalue of A, it has algebraic multiplicity 1, and each eigenvector corresponding to Δ is of the form $c \cdot \vec{1}$ for any scalar c.*

(ii): *For every eigenvalue λ of A, $|\lambda| \leq \Delta$.*

Proof: Since A is the adjacency matrix of a Δ-regular graph, the number of 1's in every row of A is Δ, hence $cA\vec{1} = c\Delta\vec{1}$, for any scalar c, and Δ is an eigenvalue of A. This implies, by Remark 8.1.1, that the geometric multiplicity of Δ is non-zero, and hence, there exists a non-zero eigenvector x that corresponds to Δ (that is, $A\vec{x} = \Delta\vec{x}$). Let x_j be the largest coordinate in $\vec{x}$. Let A_j denote the jth row of A. We know that $A_j x = \Delta x_j$. However, in A_j there are 1's only in the Δ positions that correspond to the neighbors of j. Let $N(j)$ denote the set of neighbors of j in the graph. Therefore:

$$\sum_{i \in N(j)} x_i = \Delta x_j.$$

Since x_j is maximal, we conclude that $x_i = x_j$ for every i that is a neighbor of j. We inductively continue with each x_i (i is a neighbor of j) to get that all coordinates of $\vec{x}$ are equal. Therefore, $\vec{x} = c\vec{1}$ for some c, and this proves that the geometric multiplicity of Δ is 1.

Let λ be an eigenvalue of A, $\vec{x}$ an eigenvector corresponding to λ, and x_j the largest coordinate in $\vec{x}$. Since $A\vec{x} = \lambda\vec{x}$, we get

$$\sum_{i \in N(j)} x_i = \lambda x_j.$$

Taking the absolute value of both sides of the last equality, and using a generalization of the triangle inequality, we obtain:

$$|\lambda||x_j| = \left| \sum_{i \in N(j)} x_i \right| \leq \sum_{i \in N(j)} |x_i| \leq \Delta |x_j|.$$

(Recall that the number of neighbors of j is Δ, and that x_j is maximal among all entries in x.) This yields that $|\lambda| \leq \Delta$. $\square$

8.2 Eigenvalues and expanders

Let G be a Δ-regular graph, and let A be its adjacency matrix. We know that $A = A^T$, and that we can find an orthonormal basis $\vec{e}_1, \ldots, \vec{e}_n$ of eigenvectors of A. Let $\lambda_1, \ldots, \lambda_n$ be eigenvalues of A, where $\vec{e}_i$ corresponds to λ_i, for every i. We assume that $\lambda_1 \geq \cdots \geq \lambda_n$, and that $\Delta = \lambda_1$ (we already know that Δ is an eigenvalue of A), and we fix the corresponding $\vec{e}_1$ to be $\frac{1}{\sqrt{n}}\vec{1}$.

Given a set of vertices S of size bn, $(0 < b < 1)$, define the vector $\vec{z}$ to be:

$$z_u = \begin{cases} 1-b & \text{if } u \in S, \\ -b & \text{otherwise} \end{cases}$$

for every $u \in V$. Note that

$$\sum_{u \in V} z_u = bn(1-b) + (1-b)n(-b) = 0.$$

Therefore, the inner product of $\vec{z}$ and $\vec{e}_1$, $\langle \vec{z}, \vec{e}_1 \rangle$, is 0, implying that $\vec{z}$ and $\vec{e}_1$ are orthogonal. Since the $\vec{e}_i$'s form a basis, we can represent $\vec{z}$ as:

$$\vec{z} = \sum_{i=1}^{n} \alpha_i \vec{e}_i,$$

for some real numbers $\alpha_1, \ldots, \alpha_n$. Clearly, $\alpha_1 = 0$. Consider now the vector $A\vec{z}$. We have,

$$A\vec{z} = A \sum_{i=2}^{n} \alpha_i \vec{e}_i = \sum_{i=2}^{n} \alpha_i A \vec{e}_i = \sum_{i=2}^{n} \alpha_i \lambda_i \vec{e}_i,$$

and:

$$\begin{aligned} \|A\vec{z}\|^2 &= \langle A\vec{z}, A\vec{z} \rangle \\ &= \sum_{i=2}^{n} \alpha_i^2 \lambda_i^2 \|\vec{e}_i\|^2 = \sum_{i=2}^{n} \alpha_i^2 \lambda_i^2 \qquad (8.1) \\ &\leq \sum_{i=2}^{n} \alpha_i^2 \lambda_2^2 = \lambda_2^2 \|\vec{z}\|^2 \\ &= \lambda_2^2((1-b)^2 bn + (1-b)nb^2) = \lambda_2^2(1-b)bn. \end{aligned}$$

Equality (8.1) follows from the fact that $\|\vec{e}_i \vec{e}_j\| = 0$, for $i \neq j$, and $\|\vec{e}_i \vec{e}_i\| = 1$. On the other hand:

$$\|A\vec{z}\|^2 = \sum_{u \in V} ((N_S(u)(1-b) + (\Delta - N_S(u))(-b))^2 = \sum_{u \in V} (N_S(u) - b\Delta)^2.$$

Thus, the following theorem is immediate.

Theorem 8.2.1 *For any set $S \subseteq V$ of vertices of cardinality bn, it holds:*

$$\frac{1}{n} \sum_{u \in V} (N_S(u) - b\Delta)^2 \leq \lambda_2^2 (1-b) b.$$

Remark 8.2.1 *Notice that in a random Δ-regular graph, (where each vertex v has $b\Delta$ neighbors in S on the average), $\lambda_2^2(1-b)b$ is an upper bound on the variance of $N_S(v)$, when viewed as a random variable of v. This bound depends only on $|S|$ and not on the subset S. Note also that $\lambda_2^2(1-b)b \leq \lambda_2^2/4$.*

We are now ready to find our first bound on the *expansion* of S, that is the number of vertices not in S that have a neighbor in S, divided by the cardinality of S. Let t be the number of vertices not in S, that do not have a neighbor in S. For each vertex u of these t vertices, we have $N_S(u) = 0$. Therefore,

$$\frac{1}{n}\sum_{u\in V}(N_S(u) - b\Delta)^2 \geq \frac{tb^2\Delta^2}{n}.$$

Combining the latter bound with Theorem (8.2.1), we obtain a bound on t, as follows.

Corollary 8.2.1

$$\frac{t}{n} \leq \left(\frac{\lambda_2}{\Delta}\right)^2 \cdot \frac{1-b}{b}$$

The above bound relates the expansion of S with the ratio between the first and second eigenvalues of A. If λ_2/Δ is small, then S has many neighbors in $V - S$. Another bound is given by the following theorem, due to Tanner [Ta].

Theorem 8.2.2 *Let S be a set of vertices, and let $\Gamma(S)$ be the set of vertices that are either in S or have a neighbor in S. Then*

$$|\Gamma(S)| \geq \frac{\Delta^2\,|S|}{(\Delta^2 - \lambda_2^2)\frac{|S|}{n} + \lambda_2^2}.$$

Let $\vec{x}$ be the characteristic vector of S. Observe that $\vec{x} = \vec{z} + b\vec{1}$ and

$$\Gamma(S) = \{u \in V | (A\vec{x})_u > 0\}.$$

In order to prove the theorem, we prove the following lemma. The proof of Theorem 8.2.2 follows immediately from the combination of the two bounds stated in the lemma.

Lemma 8.2.1 *The following bounds hold:*

$$\begin{aligned} |\Gamma(S)| &\geq \frac{\Delta^2\,|S|^2}{\|A\vec{x}\|^2}. \\ \|A\vec{x}\|^2 &\leq \frac{(\Delta^2-\lambda_2^2)\,|(|\,S)^2}{n} + \lambda_2^2\,|S|\,. \end{aligned}$$

Proof: By definition of a norm of a vector, we have:

$$\|A\vec{x}\|^2 = \sum_{u\in V}(A\vec{x})_u^2.$$

We now use Jensen's Inequality that states that for positive numbers $a_1,\ldots,a_k$, and a convex function f,

$$\frac{1}{k}\sum_{i=1}^{k} f(a_i) \geq f\left(\frac{\sum_{i=1}^{k} a_i}{k}\right)$$

Thus, we have:

$$\begin{aligned}\sum_{v\in V}(A\vec{x})_u^2 &= |\Gamma(S)|\sum_{u\in V}\frac{(A\vec{x})_u^2}{|\Gamma(S)|}\\ &\geq |\Gamma(S)|\sum_{u\in\Gamma(S)}\frac{(A\vec{x})_u^2}{|\Gamma(S)|} &(8.2)\\ &\geq |\Gamma(S)|\left(\frac{\sum_{u\in\Gamma(S)}(A\vec{x})_u}{|\Gamma(S)|}\right)^2 &(8.3)\end{aligned}$$

Observe that since A is the adjacency matrix of a Δ-regular graph, and since $\vec{x}$ is the characteristic vector of S, for every $u\in S$, $(A\vec{x})_u=\Delta$. In addition, for every $u\notin S$, $(A\vec{x})_u=0$ (hence, we can write "=" in Inequality (8.2)). Using these two facts in the summation of Inequality (8.3), we get:

$$|\Gamma(S)|\left(\frac{\sum_{u\in\Gamma(S)}(A\vec{x})_u}{|\Gamma(S)|}\right)^2 = \frac{\Delta^2\,|S|^2}{|\Gamma(S)|}$$

This proves the first bound of the lemma.

Consider the representation of $\vec{x}$ as a linear combination over the basis $\vec{e}_1,\ldots,\vec{e}_n$. Let this representation be:

$$\vec{x}=\sum_{i=1}^{n}\beta_i\vec{e}_i.$$

On the other hand, we know that $\vec{x}=\vec{z}+b\vec{1}$. We remind the reader that $\vec{e}_1=\frac{1}{\sqrt{n}}\vec{1}$. Therefore, we get:

$$\vec{z}+b\vec{1}=\sum_{i=1}^{n}\beta_i\vec{e}_i=\frac{\beta_1}{\sqrt{n}}\vec{1}+\sum_{i=2}^{n}\beta_i\vec{e}_i.$$

However, since $\vec{z}$ and $\vec{1}$ are orthogonal, we obtain $\beta_1=b\sqrt{n}$, and $\beta_i=\alpha_i$, for $i=2,\ldots,n$. Consider, again, the norm of $A\vec{x}$.

$$\|A\vec{x}\|^2 = \left\|A(\sum_{i=1}^{n}\beta_i\vec{e}_i)\right\|^2=\left\|\sum_{i=1}^{n}\beta_iA\vec{e}_i\right\|^2=\left\|\sum_{i=1}^{n}\beta_i\lambda_i\vec{e}_i\right\|^2$$

$$\begin{aligned}
&= \sum_{i=1}^{n} \lambda_i^2 \beta_i^2 = \Delta^2 b^2 n + \sum_{i=2}^{n} \lambda_i^2 \alpha_i^2 \\
&\leq \Delta^2 b^2 n + \lambda_2^2 \sum_{i=2}^{n} \alpha_i^2 \\
&= \Delta^2 b^2 n - \lambda_2^2 b^2 n + \lambda_2^2 \sum_{i=1}^{n} \beta_i^2 \\
&= (\Delta^2 - \lambda_2^2) b^2 n + \lambda_2^2 |S| \qquad (8.4) \\
&= (\Delta^2 - \lambda_2^2) |S|^2 / n + \lambda_2^2 |S|
\end{aligned}$$

Equality (8.4) follows from the fact that

$$|S| = \vec{x}^T \vec{x} = \sum_{i=1}^{n} \beta_i^2 \|\vec{e}_i\|^2 = \sum_{i=1}^{n} \beta_i^2$$

This proves the second bound of the lemma. □

A graph $G = (V, E)$, $|V| = n$, is called an *expander*, with an *expansion parameter* $c > 0$, if for every set S of vertices of size at most $n/2$:

$$|\Gamma(S)| \geq (1 + c(1 - |S|/n))\, |S|,$$

where $\Gamma(S)$ is the set of vertices that are either in S, or have a neighbor in S. Note that the *expansion*, (that is $|\Gamma(S) - S| \,/\, |S|$), is at least $c/2$.

We now find the expansion parameter given by Tanner's bound. Let $\rho = |\lambda_2|/\Delta$. By using Tanner's theorem and the definition of an expander, we get:

$$\frac{|S|}{(1-\rho^2)|S|/n + \rho^2} = (1 + c(1 - |S|/n))|S|.$$

Substituting $b = |S|/n$ gives us:

$$c = \frac{1-\rho^2}{(1-\rho^2)b + \rho^2}.$$

For $b = 1/2$, we get

$$c = \frac{1-\rho^2}{1+\rho^2}.$$

Corollary 8.2.2 *Any Δ-regular graph is an expander with expansion parameter c satisfying*

$$c \geq \frac{1-\rho^2}{1+\rho^2}.$$

8.3 An explicit construction

We now give an explicit construction of an expander. The analysis of the construction is rather complicated and is beyond the scope of the course. The construction is due to Lubotzky, Phillips, and Sarnak [LPS], and Margulis [Ma].

Let n be some parameter, and let p and q be two distinct prime numbers, such that

$$p = 1 \pmod{n}, \text{ and } q = 1 \pmod{n}.$$

In addition, p must be a *quadratic residue* in the field GF$[q]$, that is, there exists y such that

$$y^2 = p \pmod{q}.$$

The vertices V of the graph G correspond to the set of all 2×2 matrices in the field GF$[q]$ whose determinant is 1. The matrices A and $-A$ correspond to the same vertex. There is an edge (A, B), if AB^{-1} is of the form

$$\pm\frac{1}{\sqrt{p}}\begin{pmatrix} a_0 + ia_1 & a_2 + ia_3 \\ -a_2 + ia_3 & a_0 - ia_1 \end{pmatrix},$$

where $i^2 = 1 \pmod{p}$, a_0 is odd, a_1, a_2, and a_3 are even, and

$$a_0^2 + a_1^2 + a_2^2 + a_3^2 = p.$$

Jacobi proved that there are $p+1$ sets of a_i's that match the above requirements, so the degree of the LPS graph is $p+1$. An analysis shows that,

- The number of vertices is $q(q^2 - 1)/2$,
- $\Delta = p + 1$, and
- $\lambda_2 \leq 2\sqrt{p} \approx 2\sqrt{\Delta}$.

Using Corollary 8.2.2 we obtain that the LPS graph has an expansion parameter that is at least

$$\frac{1 - (\lambda_2/\Delta)^2}{1 + (\lambda_2/\Delta)^2} \geq \frac{\Delta - 4}{\Delta + 4} = \frac{p-3}{p+5}.$$

An interesting property of the LPS expander is that it is "locally" explicit. This, and related notions, are discussed in detail in Section 18.4. We just mention here that this means that there exists an algorithm for testing whether two vertices in the expander are adjacent, and whose running time is bounded by a polynomial in the logarithm of V, the number of vertices. This property is important in some applications, e.g., Section 10.4.1.

Chapter 9

Random Walks

9.1 Basics of Markov chains

Let I denote a state space. A discrete parameter Markov chain with stationary transition probabilities is a sequence of discrete random variables $\{X_k : k \geq 0\}$, possessing the following property: for every $k \geq 1$, and every $i_0, \ldots, i_k$ in the state space I, we have

$$\begin{aligned} \Pr[X_k = i_k | X_{k-1} = i_{k-1}, X_{k-2} = i_{k-2}, \ldots, X_0 = i_0] = \\ \Pr[X_k = i_k | X_{k-1} = i_{k-1}] = p_{i_{k-1} i_k} \ . \end{aligned} \tag{9.1}$$

For every pair $(i, j) \in I^2$, the probability p_{ij} is called the *(one-step) transition probability from i to j*. The matrix $P = (p_{ij})_{i,j \in I}$ is called the *transition matrix*. We assume in the sequel that the state space I is finite.

Let us now introduce the *k-step transition probabilities* $p_{ij}^{(k)}$. We define

$$p_{ij}^{(k)} = \Pr[X_{m+k} = j | X_m = i]$$

(due to (9.1)), this quantity is the same for every $m \geq 0$). The *Chapman-Kolmogorov Equation* states that

$$p_{ij}^{(k+1)} = \sum_{l \in I} p_{il}^{(k)} p_{lj} \ ,$$

Introducing the matrix $P^{(k)} = (p_{ij}^{(k)})$, we can write

$$P^{(k+m)} = P^{(k)} P^{(m)} = P^k P^m \ .$$

The matrix $P^{(k)}$ can be viewed as the k-step transition matrix.

A state i is called *accessible* from a state j if there exists a state $k > 0$ such that $p_{ij}^{(k)} > 0$. If i is accessible from j and j is accessible from i then we say that i and j *communicate*. It is possible to partition the set of states into equivalence classes such that two states belong to the same class if and only if they communicate. If there is only one class in such a partition, the Markov chain is called *irreducible*.

Let f_{ii}, $i \in I$, be the probability that the chain will return to state i at least once, given it starts from i. If $f_{ii} = 1$, then state i is *recurrent*, otherwise it is *transient*. Obviously, if $f_{ii} = 1$, the chain will return to i infinitely often. If $f_{ii} < 1$, then each time the chain is in state i, with probability $(1 - f_{ii})$, it will never return to i. The number of visits to state i is distributed geometrically with parameter f_{ii}, that is,

$$\Pr[\text{The number of visits to } i \text{ is equal to } k] = f_{ii}^k(1 - f_{ii}) \ .$$

Thus, the expectation of the number of visits to i, while starting from i, is

$$\sum_{k=1}^{\infty} \Pr[X_k = i | X_0 = i] = \sum_{k=1}^{\infty} p_{ii}^{(k)} \ .$$

If $\sum_{k=1}^{\infty} p_{ij}^{(k)} < \infty$, then i is transient, otherwise it is recurrent. It is easy to prove that if a Markov chain is irreducible, then all states are recurrent.

Denote by H_{ij} the expected number of steps needed to visit state j for the first time, when starting from state i. If $H_{ii} < \infty$, then i is called *non-null recurrent*, otherwise it is called *null recurrent*.

Claim 9.1.1 *All states in an irreducible Markov chain are non-null recurrent.*

If state i communicates with itself, then the greatest common divisor of the set of positive k-s, such that $p_{ij}^{(k)} > 0$, is called the *period* of i and is denoted by d_i. If $d_i = 1$, then i is *aperiodic*.

A finite Markov chain where all states are non-null recurrent is called *ergodic*.

In what follows, we consider only ergodic Markov chains.

Theorem 9.1.1 (The Ergodic Theorem) *Suppose $\{X_k : k \geq 0\}$ is an ergodic Markov chain with the set of states $I = \{1, , \ldots, n\}$. Then,*

- *There exists a vector* $\pi = (\pi_1, \ldots, \pi_n)$ *such that for every* $j \in I$ *there holds*

$$\pi_j = \lim_{k \to \infty} p_{ij}^{(k)}$$

(The limit value does not depend on i*);*

- *The vector* π *is the only non-negative solution of the system of linear inequalities*

$$\begin{aligned} \pi_j &= \sum_{i=1}^{n} \pi_i p_{ij} \\ \sum_{j=1}^{n} \pi_j &= 1 \ ; \end{aligned}$$

- *For every* i, $1 \leq i \leq n$, *it holds that* $f_{ii} = 1$ *and* $H_{ii} = 1/\pi_i$ *;*

- *Denote by* $N(i,t)$ *the number of visits to state* i *within the* t *first steps. Then*

$$\lim_{t \to \infty} \frac{N(i,t)}{t} = \pi_i \ .$$

9.1.1 Random walks on graphs

Let $G = (V, E)$ be a finite, undirected, connected, and non-bipartite graph, where $|V| = n$. Define a *random walk on* G to be a Markov chain where the set of states is V, and the transition matrix is,

$$p_{ij} = \begin{cases} 1/d(i), & (i,j) \in E(G) \\ 0, & (i,j) \notin E(G) \ , \end{cases}$$

where $d(i)$ is the degree of vertex i.

Theorem 9.1.2 *Consider the above defined Markov chain. Then, for every* $1 \leq i \leq n$, *it holds:*

$$\pi_i = \frac{d(i)}{2m} \ ,$$

where $m = |E(G)|$.

Proof: It suffices to verify that the vector π defined above satisfies the matrix equation $\pi P = \pi$. Indeed,

$$\sum_{i=1}^{n} \pi_i p_{ij} = \sum_{(i,j)\in E(G)} \frac{d(i)}{2m} \cdot \frac{1}{d(i)} = \frac{d(j)}{2m} = \pi_j$$

□

Corollary 9.1.1 *For every $1 \le i \le n$, it holds:*

$$H_{ii} = \frac{1}{\pi_i} = \frac{2m}{d(i)}$$

9.2 Electrical networks and random walks

9.2.1 Preliminaries

Suppose we are given a finite, connected graph $G = (V, E)$. Let each edge $e \in E$ contain a resistor with resistance R_e. We denote by i_v the current which enters (and exits) the vertex $v \in V$. We would like to compute the potentials at each vertex $v \in V$, and the currents on each edge $e \in E$.

By Kirchhoff's Laws, the current flowing into v must be equal to the current flowing out of v. By Ohm's Law, if vertices u and v are connected by an edge $e = (u, v)$, with resistance R_e, then the current i_{uv} that flows from u to v is equal to

$$i_{uv} = \frac{\phi(u) - \phi(v)}{R_e} \ ,$$

where $\phi(\cdot)$ denotes the potential of a vertex.

In order to compute the potentials and currents in G, we write down the system of linear equations with variables i_{uv}, for all $(u, v) \in E$, according to the following rules:

1. By Kirchhoff's Laws, for every vertex $v \in V$, the sum of all currents, plus the external current (if it exists), is equal to zero. (There are n such equations);

2. The potential of every cycle in G is equal to zero.

It is well known that the set of cycles of G forms a vector space $\mathcal{C}$ with respect to symmetric difference. This vector space is of dimension $(m-n+1)$, since for every spanning tree T of G, the set of fundamental cycles, (those obtained by adding an edge $e \in E(G) \setminus E(T)$ to T), forms a basis of $\mathcal{C}$. Clearly, it suffices to verify that Requirement (2) only holds for each cycle in the basis, and this gives us $m - n + 1$ additional equations. Altogether, the two above requirements result in $(m + 1)$ equations with m variables. This system can be proved to have a non-trivial solution.

9.2.2 The cover time

For every pair of vertices $u \neq v$, define the *commute time* from u to v, C_{uv}, to be the expected number of steps to go from vertex u to vertex v, and back to vertex u, i.e., $C_{uv} = H_{uv} + H_{vu}$. Denote by C_u, $u \in V(G)$, the expected number of steps required to visit all the vertices of G, starting from u. We define the *cover time* of G to be

$$C_G = \max_{u \in V(G)} C_u$$

Suppose that every edge $e \in E$ has resistance $R_e = 1$. Suppose further that a current of unit magnitude is injected at each vertex $u \in V$, and a current of unit magnitude leaves each vertex $v \in V$. Denote by R_{uv} the effective resistance between u and v. Note that R_{uv} is equal to the potential difference $\phi(u) - \phi(v)$.

Theorem 9.2.1 $C_{uv} = 2mR_{uv}$.

Proof: Let the set of neighbors of each vertex $v \in V$ be denoted by $\Gamma(v) = \{u \in V : (u, v) \in E\}$, $d(v) = |\Gamma(v)|$. Define Φ_u^v to be the potential of u when $d(x)$ units of current enter each vertex $x \in V$, and $2m$ units of current exit v. We need the following claim.
Claim: $H_{uv} = \Phi_u^v$.
Proof: For every edge $(u, w) \in E$, by Ohm's Law:

$$\Phi_u^v - \Phi_w^v = i_{uw} R_{uw} = i_{uw}$$

Kirchhoff's Law implies that for every $u \in V - v$:

$$d(u) = \sum_{w \in \Gamma(u)} \Phi_u^v - \Phi_w^v \tag{9.2}$$

On the other hand, it is quite obvious from the definition of H_{uv}, $(u, v) \in E(G)$, that for every $u \in V - v$:

$$H_{uv} = \sum_{w \in \Gamma(u)} \frac{1}{d(u)} (1 + H_{wv}) \tag{9.3}$$

It is easy to see that systems (9.2) and (9.3) are in fact the same system of linear equalities, and hence they have the same solution, that is, $H_{uv} = \Phi_u^v$ for every $u, v \in V$.

We continue with the proof of the theorem. Consider the following network: a current of magnitude $2m$ enters u and a current of magnitude $d(x)$ exits every vertex $x \in V(G)$. If the potential of u in this network is assumed to be equal to zero, then the potential of v is equal to $-H_{uv}$. Perform now a superposition of this network with the network considered in the above claim. In the resulting network, all external currents cancel, except for those in vertices u (where the current of magnitude $2m$ enters) and v (where the current of magnitude $2m$ exits). The difference of potentials between u and v will be $H_{uv} - (-H_{vu}) = H_{uv} + H_{vu} = C_{uv}$. Hence,

$$C_{uv} = 2mR_{uv}$$

$\square$

Proposition 9.2.1 *For every pair of vertices u and v, the effective resistance R_{uv} is not more than* $\mathrm{dist}_G(u, v)$.

Corollary 9.2.1 (1) *If $(u, v) \in E(G)$, then $C_{uv} \leq 2m$;*
(2) *For every pair $u, v \in V(G)$:*

$$C_{uv} \leq 2m(n-1) \ .$$

We can now obtain the following upper bound on the cover time of a graph.

Theorem 9.2.2 $C_G \leq 4m(n-1)$.

Proof: Let T be a spanning tree of G rooted at u. Any search algorithm on T, (DFS for example), can easily be converted to a directed path, having $2(n-1)$ edges, starting from u, visiting all vertices of G and returning to u. For every edge (u, v) of this directed path, we have $C_{uv} \leq 2m$, so $C_u \leq 2m \cdot 2(n-1) = 4m(n-1)$. Hence,

$$C_G = \max_{u \in V} C_u \leq 4m(n-1)$$

$\square$

9.2.2.1 Special cases of graphs

A complete graph K_n:

Here, the problem of calculating the cover time, is essentially the Coupon Collector problem. Therefore,

$$C_{K_n} = n \log n + cn \ .$$

A star S_n:

Again, the problem of calculating the cover time, is essentially the Coupon Collector problem. Therefore,

$$C_{S_n} = 2n \log n + cn \ .$$

A line L_n:

Let u_1, u_n be the endpoints of L_n. Then, since $R_{u_1,u_n} = n - 1$, we obtain

$$C_{u_1,u_n} = 2|E(L_n)| \cdot R_{u_1,u_n} = 2(n-1)(n-1) = 2(n-1)^2$$

By symmetry,

$$H_{u_1,u_n} = (n-1)^2$$

Then

$$C_{L_n} = C_{u_1} = H_{u_1,u_n} = (n-1)^2$$

A lollipop graph:

Suppose n is even. The *n-lollipop graph* is a complete graph on $n/2$ vertices, one of which, call it u, is attached to a line on $n/2$ vertices. Suppose that a random walk starts at u. How can we estimate C_u?

The probability of going from u to the attached line is $O(1/n)$. If we go from u to the complete graph, the expected number of steps needed to return to u is $O(n)$. (This can easily be obtained by computing the stationary probabilities of a complete graph). The expected time to exit from u to the attached line is therefore $O(n)/O(1/n) = O(n^2)$.

Consider a random walk on a line of length $n/2$, with endpoints u_1 and $u_{n/2}$. (To make the random walk ergodic, we add a loop at u_1 and $u_{n/2}$). As explained above,

$$C_{u_1,u_{n/2}} = 2\left(\frac{n}{2} - 1\right)^2$$

And by symmetry,

$$H_{u_1,u_{n/2}} = O(n^2)$$

Let us estimate the expected number of times the random walk returns to u_1 before reaching $u_{n/2}$. First, we give an intuitive explanation, which will be justified a bit later by a rigorous argument. Since the stationary probability of u_1 in the random walk on the line is $O(1/n)$, we obtain $H_{u_1,u_1} = O(n)$. The expected number of returns to u_1 before reaching $u_{n/2}$ can be estimated to be $H_{u_1,u_{n/2}}/H_{u_1,u_1} = O(n)$. Then, in the random walk on the lollipop graph,

$$H_{u,u_{n/2}} = O(n^3) \ .$$

But, we have proven that $C_{u,u_{n/2}} = 2mR_{u,u_{n/2}} \leq cn^3$. Therefore

$$C_u = \Theta(n^3) \ .$$

Let us justify our estimate of the expected number of returns to u_1 before reaching $u_{n/2}$. To this end, define a random variable Z_i to be the number of steps between the $(i-1)$st and the ith return to u_1. Then, $\mu = E[Z_i] = O(n)$, for all i. Let the random variable T count the number of returns to u_1 before reaching $u_{n/2}$. It is immediate that

$$E\left[\sum_{i=1}^{T} Z_i\right] = H_{u_1,u_{n/2}} = O(n^2)$$

It follows from Definition 1.5.1 that T is a stopping rule with respect to the sequence $\{Z_i\}$, since the event $T = i$ depends only on $Z_1, Z_2, \ldots, Z_{i-1}$. Hence, Wald's Equation, (see Theorem 1.5.1), can be used in this case, implying that

$$E\left[\sum_{i=1}^{T} Z_i\right] = \mu E[T]$$

Hence, $E[T] = O(n)$.

9.2.2.2 The effective resistance of a graph

The *effective resistance* $R(G)$, (also denoted by R), of a graph G is defined as

$$R(G) = \max_{(u,v) \in V(G)} R_{uv}$$

Theorem 9.2.3 *$C_G \geq m \cdot R(G)$.*

Theorem 9.2.4 *Suppose that $R(G) = R_{uv}$ for some pair of vertices $u, v \in V$. Then*

$$C_G \geq \max(H_{uv}, H_{vu}) \geq \frac{C_{uv}}{2} = \frac{2m \cdot R(G)}{2} = m \cdot R(G)$$

$\square$

Theorem 9.2.5 $C_G \leq 5mR\log_2 n$.

Proof: Consider the first $5mR\log_2 n$ steps of a random walk on G, and partition them into $\log_2 n$ intervals, each of length $5mR$. Fix a vertex $v \in V$. For every other vertex $x \in V$, $H_{xv} \leq 2mR$. Therefore, due to Markov's Inequality, the probability that v is not visited within an interval of length $5mR$, is at most $2/5 < 1/2$. This yields that the probability that v is never visited within $5mR\log_2 n$ steps is strictly less than $(1/2)^{\log_2 n} = 1/n$. Therefore, the probability that some vertex of G is not visited within $5mR\log_2 n$ steps is less than 1, and we conclude that there *exists* a random walk of length $5mR\log_2 n$, visiting all vetices of G.

For every subset $A \subseteq V$, the expected number of vertices of A that are visited within $5mR$ steps is more than $|A|/2$. Define A_i, $i = 0, 1, 2, \ldots$, to be the set of vertices that were not yet visited after finishing the ith interval. Obviously,

$$\begin{aligned} |A_0| &= n \ , \\ E[|A_i|] &\leq \frac{|A_{i-1}|}{2} \ . \end{aligned}$$

We need to show that the expected number of intervals of length $5mR$ needed to exhaust the vertex set V is at most $\log n$. To this end, define a random variable,

$$T(n) = \min\{k : \text{ all n vertices were visited within the first k intervals of length } 5mR\}$$

Define also n' to be a random variable counting the number of vertices that were not visited after the first interval. Then

$$\begin{aligned} T(n) &\leq 1 + T(n') \ , \\ E[n'] &\leq \frac{n}{2} \ , \\ T(1) &= 1 \ , \\ T(0) &= 0 \ . \end{aligned}$$

Claim: $E[T(n)] \leq 1 + \log_2 n$.

Proof: Define a continuous concave function

$$\Lambda(x) = \begin{cases} 1 + \log_2 x, & x > 1 \\ x, & 0 \leq x \leq 1 \end{cases}$$

We prove by induction on n that:

$$E[T(n)] \leq \Lambda(n) \ .$$

Let $p_i = \Pr[n' = i]$. Then

$$E[n'] = \sum_{i=0}^{n} i p_i \ .$$

By the induction hypothesis and by Jensen's inequality,

$$\begin{aligned} E[T(n)] &\leq 1 + E[T(n)] = 1 + \sum_{i=0}^{n} p_i T(i) \leq 1 + \sum_{i=0}^{n} p_i \Lambda(i) \\ &\leq 1 + \Lambda\left(\sum_{i=0}^{n} i p_i\right) = 1 + \Lambda(E[n']) = 1 + \Lambda\left(\frac{n}{2}\right) \\ &= 1 + \left(1 + \log_2 \frac{n}{2}\right) = 1 + \log_2 n = \Lambda(n) \end{aligned}$$

□

9.2.2.3 Rayleigh's monotonicity law

Rayleigh's monotonicity law states that the effective resistance of a graph is non-increased (resp., non-decreased), whenever the resistance of any edge of the graph is decreased (resp., increased). In particular, the effective resistance can only decrease when we add more edges to the original graph.

Claim 9.2.1 *The effective resistance of a graph is not more than its diameter, denoted by diam(G).*

Proof: The whole graph can be generated by adding edges to the subgraph that corresponds to the diameter. □

Let G be a k-regular graph on n vertices.

Theorem 9.2.6 $C_G \leq O(n^2 \log n)$.

(**Remark:** [KLNS] proved a tighter bound, i. e., $C_G \leq O(n^2)$.)

Proof: Let us first prove that

$$\mathrm{diam}(G) \leq \frac{3n}{k}.$$

Consider the shortest path $L(u,v)$ between vertices u and v, corresponding to the diameter. Clearly, any two vertices $x, y \in V(L)$, for which $\mathrm{dist}_L(x,y) = 3$, are not adjacent. Hence,

$$n \geq k \cdot \frac{|V(L)|}{3} = \frac{k \cdot \mathrm{diam}(G)}{3} \; .$$

By Theorem 9.2.3

$$C_G \leq O(mR\log n) \; ,$$

That is,

$$C_G \leq O\left(\frac{nk}{2} \cdot \frac{3n}{k} \cdot \log n\right)$$

$\square$

Chapter 10

Convergence of Random Walks

Consider the following problem: how many steps are needed for a random walk that starts at some designated vertex, to converge to the stationary distribution?

In the following sections we present three basic techniques for analyzing how rapidly a random walk approaches the stationary distribution. The first technique calculates the rate of convergence directly by defining a *stopping rule.* The second technique is called *coupling.* It reduces the problem of finding the rate of convergence to that of calculating the number of steps needed to meet another, imaginary random walk, that starts at the stationary distribution. The third technique uses methods from Linear Algebra, in particular eigenvalues, to bound the rate of convergence. This method is useful for giving a tight upper bound on the rate of convergence of a random walk in an important class of graphs called expanders.

For more details on the first two techniques, the reader is referred to [Di, Li].

10.1 Preliminaries

Recall that the *variation distance* between two probability distributions, P and Q, on a set I of states is:

$$\|P - Q\| = \max_{I' \subseteq I} |P(I') - Q(I')|.$$

Lemma 10.1.1 *Let P and Q be two probability distributions over a set I of states. Then*

$$\|P - Q\| = \frac{1}{2} \sum_{i \in I} |P(i) - Q(i)|. \tag{10.1}$$

Proof: By the definition of $\|P-Q\|$, there exists a set $M \subseteq I$, for which

$$|P(M)-Q(M)| \geq |P(I')-Q(I')|,$$

for all $I' \subseteq I$. Without loss of generality, assume that $P(M) \geq Q(M)$. We claim that for every $i \in M$, $P(i) \geq Q(i)$. Otherwise, $|P(M-\{i\})-Q(M-\{i\})| > |P(M)-Q(M)|$. Similarly, for every $i \in I-M$, $P(i) \leq Q(i)$. Having these two facts in mind, we have on the right hand side of (10.1):

$$\begin{aligned}
\frac{1}{2}\sum_{i\in I}|P(i)-Q(i)| &= \frac{1}{2}\left(\sum_{i\in M}|P(i)-Q(i)| + \sum_{i\in I-M}|P(i)-Q(i)|\right) \\
&= \frac{1}{2}\left(\sum_{i\in M}P(i)-Q(i) + \sum_{i\in I-M}Q(i)-P(i)\right) \\
&= \frac{1}{2}\left(P(M)-Q(M)+Q(I-M)-P(I-M)\right) \\
&= \frac{1}{2}\left(P(M)-Q(M)+1-Q(M)-1+P(M)\right) \\
&= P(M)-Q(M).
\end{aligned}$$

Which proves the claim. □

10.2 Strong uniform time

Recall the definition of a *stopping rule* (Definition 1.5.1): given random variables $Z_1, Z_2, \ldots$, a random variable T, whose domain is the natural numbers, is a stopping rule, if, for every i, the event $T=i$ is independent of the random variables Z_j, for all $j>i$. For a given finite ergodic Markov chain, a *strong uniform time* T, is a stopping rule which satisfies:

$$\Pr[X_k = i \mid T = k] = \pi_i,$$

where X_k is the state at the kth step in the Markov chain, and π_i is the probability, under the stationary distribution, of being at state i.

The following theorem relates strong uniform time with the rate of convergence of a random walk.

Theorem 10.2.1 *Let π be the stationary probability distribution of a random walk, and let $Q^{(t)}$ be the probability distribution after t steps. In addition, let T be a strong uniform time. Then*

$$\|Q^{(t)}-\pi\| \leq \Pr[T>t].$$

Proof: Let X_t be a random variable that denotes the state visited by the random walk at step t. We have the following equalities.

$$\begin{aligned}
\forall I' \subseteq I,\ Q^{(t)}(I') &= \Pr[X_t \in I'] && (10.2)\\
&= \left(\sum_{j\le t} \Pr[(X_t \in I') \cap (T = j)]\right) + \\
&\quad \Pr[(X_t \in I') \cap (T > t)] \\
&= \left(\sum_{j\le t} \Pr[X_t \in I' | T = j]\ \Pr[T = j]\right) + \\
&\quad \Pr[X_t \in I' | T > t]\ \Pr[T > t] \\
&= \left(\sum_{j\le t} \pi(I') \Pr[T = j]\right) + \\
&\quad \Pr[X_t \in I' | T > t]\ \Pr[T > t] && (10.3)\\
&= \pi(I')(1 - \Pr[T > t]) + \\
&\quad \Pr[X_t \in I' | T > t]\ \Pr[T > t] && (10.4)\\
&= \pi(I') + \Pr[T > t](\Pr[X_t \in I' | T > t] - \pi(I')). && (10.5)
\end{aligned}$$

Equality (10.2) follows from the definition of $Q(I')$. Equality (10.3) follows since T is a strong uniform time, and from the fact that once a random walk is in the stationary distribution, it stays there forever. Equality (10.4) follows from the fact that

$$\sum_{j\le t} \Pr[T = j] = \Pr[T \le t] = 1 - \Pr[T > t].$$

Since both $\Pr[X_t \in I' | T > t]$ and $\pi(I')$ are at most 1, Equality (10.5) implies

$$\forall I' \subseteq I,\ |Q^{(t)}(I') - \pi(I')| \le \Pr[T > t].$$

□

We now demonstrate how Theorem 10.2.1 can be used to calculate the rate of convergence of random walks.

10.2.1 The hypercube

In this example we calculate how fast a random walk on the hypercube converges to the stationary distribution. We remind the reader that a hypercube of dimension n is a graph in which:

- The set of vertices corresponds to the set of 2^n binary vectors of length n.
- (u, v) is an edge, if vertices u and v differ in exactly one coordinate (in other words, the *Hamming distance* between u and v is 1).

Consider the following random walk on an n-dimensional hypercube:

1. Choose a vertex uniformly among the neighbors of the currently-visited vertex.
2. With probability $\frac{1}{2}$, move to the chosen vertex. With probability $\frac{1}{2}$, stay still.

The Markov chain that results from the above random walk is ergodic. Note that if we always move to the chosen vertex (of Step 1), then the resulting Markov chain would not be ergodic: If we visit vertex v at step i, we will never visit v in step j, where $j - i$ is odd, i.e., every state would have a period of 2.

Define the stopping rule T to be the number of coordinates chosen so far (even if not all choices of coordinates yielded a move). Note that the random variable T is a strong uniform time stopping rule. Informally, this happens since we have an equal probability of being at any vertex of the hypercube. after all the coordinates are chosen.

In order to be able to use Theorem 10.2.1, we need to find $\Pr[T > t]$. However, this problem is exactly the coupon collector problem (see Section 1.6), since we may think of the coordinates as being coupons that need to be collected.

For the coupon collector problem, the expected number of trials, (i.e. random choices of coupons), needed to see all n coupons is $O(n \log n)$.This means that the expected number of steps until we "choose" all coordinates is $O(n \log n)$. Using Markov's Inequality, we get:

$$\Pr[T > t] \le \frac{E[T]}{t^2} = O\left(\frac{n \log n}{t^2}\right).$$

10.2.2 Card shuffling

Our next example deals with shuffling cards. Suppose we want to shuffle a pack of n cards, numbered $1, 2, \ldots, n$, according to the following policy: *Move the top card to a random location in the pack.* The question is, how long will it take us to reach the stationary distribution, i.e., have all cards in the pack in random order?

This problem may be viewed as a random walk in a graph with $n!$ vertices corresponding to all possible permutations of the cards. The edges of the graph are determined by the shuffling policy, i.e., there is an edge between u and v if and only if the state that corresponds to v may be reached from the state that corresponds to u, (according to the shuffling policy). Let the stopping rule T in this case be the number (or name) of the card moved from the top. For ease of notation, we denote by BOTTOM the card that was originally at the bottom of the pack (before starting to shuffle).

We claim that T is a strong uniform time. This happens since once we move BOTTOM from the top (to a random place in the pack), we are already in the stationary distribution. This can be established by showing (by induction on the number of cards below BOTTOM) that the cards below BOTTOM are in random order.

We now consider the distribution by which T behaves.

Lemma 10.2.1 *The stopping rule T behaves as a coupon collector.*

Proof: Define T_i to be the number of steps until there are i cards below BOTTOM. Since $T = T_n$, it is true that

$$T = T_1 + (T_2 - T_1) + (T_3 - T_2) + \cdots + (T_n - T_{n-1}).$$

In addition, $T_{i+1} - T_i$ has a geometric distribution with parameter $\frac{i+1}{n}$.

In coupon collection we have a similar situation. Let the random variable V_i denote the number of steps until we see i coupons, and let V be the number of steps until we see all coupons (so $V = V_n$). The similarity between V and T stems from the fact that

$$V = V_1 + (V_2 - V_1) + (V_3 - V_2) + \cdots + (V_n - V_{n-1}),$$

and that $V_{i+1} - V_i$ has a geometric distribution. $\square$

Therefore,

$$\Pr[T > t] \leq \frac{E[T]}{t^2} = O\left(\frac{n \log n}{t^2}\right).$$

10.3 Coupling

Consider a random walk X that starts at some fixed point. We want to investigate how fast X approaches the stationary distribution. The idea underlying the analysis by a *coupling* argument is to consider another (imaginary) random walk Y that starts at the stationary distribution. The two random walks are correlated; however, each is independent from its own point of view. Hence, Y always remains at the stationary distribution. The random walks are correlated in a way that, once they meet, future moves are the same. Thus, in

order to answer the above question, it suffices to calculate when do the random walks X and Y meet.
We henceforth consider three examples where a coupling argument is used.

10.3.1 The hypercube

Recall the hypercube graph from Section 10.2.1. Let a *coordinator* be a superplayer that coordinates the moves of X and Y. It does the following at each step..

1. The coordinator chooses a coordinate i uniformly from {1,...,n}. Let X_i (Y_i) be the ith bit of the vertex currently visited by X (Y).

2. If $X_i = Y_i$: with probability 1/2, both X and Y move, and with probability 1/2 both of them stay still. The move, (if any), is to the neighbor that differs in the ith bit from the currently visited vertex.

3. If $X_i \neq Y_i$: with probability 1/2, X stays still and Y moves, and with probability 1/2, Y stays still and X moves. Here, again, the move is to the vertex that differs in the ith bit from the currently visited vertex.

Note that from the point of view of both X and Y, they perform a random walk similar to the one defined in Section 10.2.1. The following lemma is easy to prove.

Lemma 10.3.1 *If $X_i = Y_i$, then it always stays as such. If coordinate i is chosen, (and $X_i \neq Y_i$), then $X_i = Y_i$ at the end of the step.*

Hence, the random variable that counts the number of steps needed for X and Y to meet, behaves as a coupon collector, and we obtain the same result as in section 10.2.1.

10.3.2 Card shuffling

Consider the card shuffling problem, this time with a different shuffling policy: *Uniformly at random, choose a card, and put it either, at the top, or at the bottom of the pack, alternately.* We use here a coupling argument by considering two packs of cards: one that is initially ordered (top card is 1, and the bottom one is n), and the other pack is in random order.

The coordinator here chooses, uniformly at random, a card i, and in both packs the card is moved to either the top or the bottom, alternately.

Again, we reach the stationary distribution, after all the cards were picked. It can be shown, by induction, that after i steps, each pack may be partitioned into three parts: top, bottom, and middle. The cards at the top and bottom parts have all been selected and appear in the same order in both packs, while the cards in the middle have not been selected so far, and appear in some arbitrary order in both packs. Hence, after all cards have been picked at least once, the two packs are in the same order. Observe that the problem of picking all cards is again similar to the coupon collector problem, and hence the expected number of steps needed to shuffle the pack is $O(n \log n)$.

10.3.3 Generating a spanning tree

Given a graph $G = (V, E)$, we want to generate, unifromly at random, a spanning tree of G. Throughout this example, for ease of exposition, we assume that the edges in a tree are directed towards a specific vertex r in the tree, called the *root* of the tree.

Consider a Markov chain in which the states correspond to all possible pairs (r, T), where T is a spanning tree of G, and r is the root of T. The number of states in the Markov chain is $|V| \cdot s$, where s is the number of spanning trees of G. We define a random walk as follows:

- Randomly choose an edge e that connects the root r to a vertex y.
- If edge e is in the tree, (and therefore directed from y to r), then change its direction, hence making y the root of the tree.
- If e is not in the tree, add e to the tree, direct it from r to y, and delete from the tree the (unique) outgoing edge leaving y.

Notice that, the indegree of each state in the Markov chain, is equal to the degree of the root (of the tree) in the graph. This follows, since a state (r, T) can be reached from all the neighbors of r in G. A similar argument also implies that, the outdegree of each state, is equal to the degree of the root (of the tree) in the graph. Hence, the probability of choosing a state (r, T) at the stationary distribution is proportional to the degree of r in G.

In order to find how fast the random walk approaches the stationary distribution, we use a coupling argument. Consider two random walks, one which starts at the stationary distribution, and the other which starts at an arbitrary state, moving with the following coordination:

Stage I: The two walks progress independently until they agree on a root vertex. (Note that the tree might still be different.)

Stage II: Once the two walks agree on a root, from now on, they make the same moves.

Notice that, if we think of the root of the tree as a particle, then it (implicitly) performs a random walk in the graph G. This observation is crucial to our analysis. We now calculate the expected number of steps until the two random walks meet, i.e., generate the same spanning tree.

Stage I: Calculating the expected number of steps of Stage I is equivalent to the following problem. Two particles are moving at random in the graph G. What is the expected number of steps until they meet?

Lemma 10.3.2 *The expected number of steps until the two particles meet is at most twice the cover time of the graph.*

Stage II: At the beginning of this stage, both spanning trees, (in the corresponding states of the two random walks), have the same root. Note that when a new root is chosen, there is an edge connecting the old root to the new root, and it exists in both trees, yielding that both spanning trees remain with the same root. Suppose the root vertex is switched from r to r'. Notice that, from now on, the (unique) outgoing edge of vertex r, will remain the same in both trees. This happens, since the outgoing edge of a vertex r changes when, either r becomes the root, or, when it ceases to be the root. It follows, that a sufficient condition for the two trees to be identical, is that each vertex in the graph is a root of the tree at some point in time. Hence, an upper bound on the time for the two trees to converge at Stage II is the cover time of G.

We conclude that the expected number of steps, in both stages, until the spanning trees become identical, linearly depends on the cover time of the graph.

10.4 Random walks on expanders

Expanders have the property of "rapidly mixing", that is, a "small" number of steps in a random walk converges to the stationary distribution. In this section we prove this fact. (The reader is advised to read Chapter 8 before reading this section).

Assume the expander is a Δ-regular bipartite graph $G = (V, E)$ (there are explicit constructions for such expanders in the literature), and let A be its adjacency matrix. In order to make the random walk on G ergodic, we add a self loop to every vertex in G. With probability $1/2$ we stay in the vertex, and with probability $1/2$ we move. The transition matrix Q, is therefore:

$$Q = \frac{I + A/\Delta}{2}$$

This means that if $\lambda_1, \ldots, \lambda_n$ are the eigenvalues of A, and $\lambda'_1, \ldots, \lambda'_n$ are the eigenvalues of Q, then

$$\lambda'_i = \frac{1 + \lambda_i/\Delta}{2}$$

for $i = 1, \ldots, n$. Note that $\lambda'_1 = 1$.

Let $\vec{q}^{\,(t)}$ denote the transition distribution at step t (that is, $q_i^{(t)}$ is the probability of being at state i after t steps). Define the variation distance after t steps to be:

$$\Delta(t) = \max_{1 \le i \le n} \frac{|q_i^{(t)} - \pi_i|}{\pi_i}$$

Next, we are going to prove a theorem, from which we conclude that expanders are rapidly mixing. Beforehand, we need one preliminary lemma.

Lemma 10.4.1 *Let $\vec{a}$ be a vector in $\Re^n$, then $|\vec{a}| \le \sqrt{n}\|\vec{a}\|$.*

Proof: We use the Cauchy-Schwartz Inequality, which states that for any two vectors $\vec{a}, \vec{b} \in \Re^n$, $|\vec{a}^T\vec{b}| \le \|\vec{a}\| \cdot \|\vec{b}\|$. Choose $\vec{b}$ to be:

$$b_i = \begin{cases} 1 & \text{if } a_i \ge 0, \\ -1 & \text{otherwise.} \end{cases}$$

For the above $\vec{b}$ we get that $|\vec{a}^T\vec{b}| = |\vec{a}|$, and $\|\vec{b}\| = \sqrt{n}$, which establishes the claim. $\square$

Theorem 10.4.1 *For every choice of $\vec{q}^{\,(0)}$, $\Delta(t) \le n^{1.5}(\lambda'_2)^t$.*

Proof: By definition of $\vec{q}^{\,(t)}$, $\vec{q}^{\,(t)} = Q^t \vec{q}^{\,(0)}$. Let $\vec{e}_1, \ldots, \vec{e}_n$ be the eigenvectors of Q, and suppose that $q^{(0)}$ is represented by:

$$\vec{q}^{\,(0)} = \sum_{i=1}^{n} c_i \vec{e}_i.$$

Without loss of generality, assume that $\pi = c_1 \vec{e}_1$, where π is the stationary distribution of the random walk. Thus, we have:

$$\vec{q}^{\,(t)} \;=\; \sum_{i=1}^{n} c_i Q^{(t)} \vec{e}_i = \sum_{i=1}^{n} c_i (\lambda_i')^t \vec{e}_i = c_1 \vec{e}_1 + \sum_{i=2}^{n} c_i (\lambda_i')^t \vec{e}_i. \tag{10.6}$$

Use now Lemma 10.4.1 to get: $|\vec{q}^{\,(t)} - c_1 \vec{e}_1| \leq \sqrt{n} \|\vec{q}^{\,(t)} - c_1 \vec{e}_1\|$. By Equation (10.6), we have:

$$\begin{aligned}
\sqrt{n}\, \|\vec{q}^{\,(t)} - c_1 \vec{e}_1\| &= \sqrt{n} \left\| \sum_{i=2}^{n} c_i (\lambda_i')^t \vec{e}_i \right\| && (10.7)\\
&= \sqrt{n} \sqrt{\sum_{i=2}^{n} c_i^2 (\lambda_i')^{2t}} && (10.8)\\
&\leq \sqrt{n} (\lambda_2')^t \sqrt{\sum_{i=2}^{n} c_i^2} && (10.9)\\
&\leq \sqrt{n} (\lambda_2')^t. && (10.10)
\end{aligned}$$

Inequality (10.9) follows from the fact that $\lambda_i \leq \lambda_2$, for every $i = 2, \ldots, n$. Inequality (10.10) follows since

$$\sum_{i=2}^{n} c_i^2 \leq \sum_{i=1}^{n} c_i^2 = \|\vec{q}^{\,(0)}\|^2 \leq 1.$$

We are ready now to establish the claim:

$$\begin{aligned}
\Delta(t) = \max_i \frac{q_i^{(t)} - \pi_i}{\pi_i} &= n \max_i |q_i^{(t)} - \pi_i| && (10.11)\\
&\leq n |\vec{q}^{\,(t)} - \pi| \leq n^{1.5} (\lambda_2')^t. && (10.12)
\end{aligned}$$

Equality (10.11) follows from the following fact:

$$\max_i |q_i^{(t)} - \pi_i| \leq \sum_i |q_i^{(t)} - \pi_i| = |\vec{q}^{\,(t)} - \pi|.$$

Inequality (10.12) follows immediately from Inequality (10.10). $\square$

Finally, we would like to determine how rapidly does the transition distribution approach the stationary distribution. Formally, given $0 < \delta < 1$, after how many steps $T(\delta)$, do we get $\Delta(t) \leq \delta$? By substituting $\delta = n^{1.5}(\lambda_2')^{T(\delta)}$, we get

$$T(\delta) = \frac{\log \frac{n^{1.5}}{\delta}}{-\log \lambda_2'}.$$

In the LPS expander that is presented in Section 8.3, $\lambda_2' \approx \frac{1}{2} + \frac{1}{2\sqrt{\Delta}}$. Hence, $\log \lambda_2'$ is bounded by a constant. Fix δ to be $1/\mathrm{poly}(n)$, for some polynomial $\mathrm{poly}(n)$. For such values of λ_2' and δ, we obtain $T(\delta) = O(\log n)$.

In other words, the random walk converges to the stationary distribution after $O(\log n)$ moves, which is optimal (up to constant factors), since the diameter of a regular graph is at least $\Omega(\log n)$.

10.4.1 An application: amplifying the probability of success

In this section we will see how to apply expander graphs to the following problem. Let $\mathcal{A}$ be a randomized algorithm that uses n random bits and has probability of success $\geq \frac{1}{2}$. In other words, $\mathcal{A}$ chooses uniformly at random a string from $\{0,1\}^n$, and we are guaranteed that at least half of the strings lead $\mathcal{A}$ to a correct solution. We would like to amplify the probability of success of $\mathcal{A}$ from $1/2$ to $1 - (\frac{1}{2})^k$, for a given number $k(n)$.

This goal can be easily achieved by running k independent trials of $\mathcal{A}$. However, the number of random bits used is kn. We now show that expanders can help us achieve this goal by using only $O(n+k)$ random bits. Thus, we can achieve an exponentially small error probability, while paying only an additive (instead of multiplicative) increase in the number of random bits. This result is due to Ajtai, Komlos and Szemeredi [AKS]. (See also [CW, IZ]).

Let $G = (V, E)$ be an expander such that each vertex in V corresponds to a (different) string over $\{0,1\}^n$. ($| V | = 2^n$). We choose G to be an expander of fixed degree. One possible choice for an expander is the LPS construction (see Section 8.3). Although the G contains an exponential number of vertices, the LPS construction has the advantage that deciding whether two vertices are adjacent can be done in polynomial time.

The Amplification Algorithm

1. Choose a vertex $v_0 \in V$ in G, uniformly at random.

2. Perform a random walk of length $O(k)$ in G. Let $v_0, v_1, \ldots$ be the vertices corresponding to the random walk.

3. Run $\mathcal{A}$ on all strings that correspond to $v_0, v_1, \ldots$ in the walk.

4. If $\mathcal{A}$ succeeds, then output the solution.

Random Bit Complexity: The complexity of Step (1) is n random bits. Each step in the random walk requires $O(1)$ bits, since the degree of G is fixed. Thus, we need a total of $O(k)$ bits for the walk, and the entire algorithm uses $O(n+k)$ bits.

To prove the correctness of the algorithm, we split G into "good" vertices, i.e., vertices that lead to a correct solution, and "bad" vertices, i.e., vertices that lead to an incorrect solution. It is clear that at least half of the vertices are "good". We need to show that the probability that a random walk of length $O(k)$ will not hit a "good" vertex is at most $(\frac{1}{2})^k$. We prove an appropriate general assertion in the following theorem.

Theorem 10.4.2 *Let $G = (V, E)$ be a Δ-regular expander. Let $C \subset V$ be a subset such that $\mid C \mid = c \cdot n$. For each l, the number of paths of length l that do not pass through C is at most $n \cdot \sqrt{1-c} \cdot [(1-c)\Delta^2 + \lambda_2^2]^{l/2}$, where λ_2 is the absolute value of the second eigenvalue of G.*

Proof: Let A be the adjacency matrix of G, and let P be its projection on the set $V \setminus C$. I.e., each column P_i in P is defined as

$$P_i = \begin{cases} e_i & \text{if } v_i \in V \setminus C \\ \vec{0} & \text{otherwise} \end{cases}$$

where e_i is the ith unit vector, i.e., $e_i = \{\overbrace{0, \ldots, 0}^{i-1}, 1, \overbrace{0, \ldots, 0}^{n-i+1}\}$. We claim that $P \cdot A$ represents a single step of a random walk in G. This happens because

$$(P \cdot A)_{ij} = \begin{cases} A_{ij} & \text{if } i \in V \setminus C \\ 0 & \text{otherwise} \end{cases}$$

Hence, $(P \cdot A)_{ij}$ contains a 1 if $i \in V \setminus C$ and there exists an edge (or a path of length 1) in G between i and j; otherwise, $(P \cdot A)_{ij}$ contains a 0. We claim that the number of paths in G of length l that are contained in $V \setminus C$ is $(P \cdot A)^l \cdot P \cdot \vec{1}$. This is true since $(P \cdot A)^l$

contains all paths of length l that are contained in $V \setminus C$, and $P \cdot \vec{1}$ is the characteristic vector of $V \setminus C$.

We need to calculate $|(PA)^l P \cdot \vec{1}|_1$. We start by calculating $||(PA)^l P \cdot \vec{1}||_2$. Recall that given a matrix B, $|| B ||_2 = \max_x || Bx ||_2$ were the maximum is taken over all vectors x such that $|| x ||_2 = 1$.

Lemma 10.4.2 $|| P \cdot A ||_2 \leq \sqrt{(1-c)\Delta^2 + \lambda_2^2}$.

Proof: Let z be a real vector such that $|| z ||_2 = 1$. We may write z as a linear combination $z = u+v$ where $u = \vec{1} \cdot$constant is an eigenvector that corresponds to the largest eigenvalue, and v is a vector such that the sum of its coordinates is 0. Thus, $u \perp v$. Therefore,

$$|| z ||_2^2 = ||u+v||_2^2 = ||u||_2^2 + ||v||_2^2 + 2||u \cdot v||_2 = ||u||_2^2 + ||v||_2^2 = 1$$

and $A \cdot u = \Delta \cdot u$. Thus,

$$P \cdot A \cdot u = \begin{cases} A \cdot u & \text{in all coordinates that corresponds to } V \setminus C \\ 0 & \text{otherwise} \end{cases}$$

Note that all coordinates in $A \cdot u$ are the same since G is a regular graph. Hence, $A \cdot u = \{\Delta \cdot \text{constant}, \Delta \cdot \text{constant}, \ldots\}$ yielding that $|| P \cdot A \cdot u ||_2 = \sqrt{1-c} \cdot \Delta \cdot || u ||_2$.

Recall that $v \perp u$. Hence, v is spanned by eigenvectors that correspond to eigenvalues smaller than Δ. Thus, $|| A \cdot v ||_2 \leq \lambda_2 \cdot || v ||_2$. We claim that $|| P \cdot A \cdot v ||_2 \leq || A \cdot v ||_2$, since P only sets to 0 some of the coordinates in $A \cdot v$. Hence, $|| P \cdot A \cdot v ||_2 \leq \lambda_2 \cdot || v ||_2$. Recall that

$$|| z ||_2 = || u + v ||_2 \leq || u ||_2 + || v ||_2 \quad \text{(Triangle Inequality)}$$

Hence,

$$|| P \cdot A \cdot z ||_2 \leq || P \cdot A \cdot u ||_2 + || P \cdot A \cdot v ||_2 \leq \sqrt{1-c} \cdot \Delta || u ||_2 + \lambda_2 \cdot || v ||_2$$

Recall that by Cauchy-Schwartz,

$$a \cdot || u ||_2 + b \cdot || v ||_2 \leq \sqrt{a^2 + b^2} \cdot \sqrt{|| u ||_2^2 + || v ||_2^2}$$

and $|| u ||_2^2 + || v ||_2^2 = 1$. Hence,

$$|| P \cdot A \cdot z ||_2 \leq \sqrt{(1-c) \cdot \Delta^2 + \lambda_2^2}$$

implying that

$$|| \ P \cdot A \ ||_2 \leq \sqrt{(1-c) \cdot \Delta^2 + \lambda_2^2}$$

□

Since $|| \ B^l \ ||_2 \leq || \ B \ ||_2^l$, it follows that

$$|| \ (P \cdot A)^l \ ||_2 \leq [(1-c) \cdot \Delta^2 + \lambda_2^2]^{l/2}$$

Since $|| \ P \cdot \vec{1} \ ||_2 = \sqrt{(1-c)n}$, (this is the number of coordinates that are not 0 in the characteristic vector of P), we have

$$|| \ (P \cdot A)^l \cdot P \cdot \vec{1} \ ||_2 \leq || \ (P \cdot A)^l ||_2 \cdot || \ P \cdot \vec{1} \ ||_2 \leq [(1-c) \cdot \Delta^2 + \lambda_2^2]^{l/2} \sqrt{(1-c)n}$$

By Cauchy-Schwartz, for any vector v, $| \ v \ |_1 \leq \sqrt{n} \cdot || \ v \ ||_2$. Hence, we have

$$| \ (P \cdot A)^l \cdot P \cdot \vec{1} \ |_1 \leq n \cdot [(1-c) \cdot \Delta^2 + \lambda_2^2]^{l/2} \sqrt{1-c}$$

which is the number of paths of length l that that do not pass through C.

Note that the total number of paths of length l in G is $n \cdot \Delta^l$. Hence, the ratio between the "bad" paths and the total number of paths is given by

$$\frac{n \cdot [(1-c) \cdot \Delta^2 + \lambda_2^2]^{l/2} \sqrt{1-c}}{n \cdot \Delta^l}$$

□

Since G is an LPS expander, $\lambda_2 \leq 2\sqrt{\Delta}$. Assuming that the probability of success is $c \geq \frac{1}{2}$, and by choosing $\Delta = 20$, we get

$$\sqrt{(1-c) \cdot \Delta^2 + \lambda_2^2} \leq \frac{\Delta}{2^{1/4}}$$

Hence,

$$\frac{n \cdot [(1-c) \cdot \Delta^2 + \lambda_2^2]^{l/2} \sqrt{1-c}}{n \cdot \Delta^l} \leq \frac{n \cdot \Delta^l \cdot 2^{-l/4}}{n \cdot \Delta^l} \leq 2^{-l/4}$$

Therefore, if we choose $l = 4k$, then the probability that all the paths generated by the algorithm, will not pass through C, is at most $(\frac{1}{2})^k$.

Part II

Randomized Algorithms

Chapter 11

Algebraic Methods

In this chapter we will see some examples of randomized algorithms that use algebraic techniques. We first show how to decide whether a polynomial given by a "black-box" is identically zero. Then we show two applications: Deciding whether two matrices are similar, and whether a graph contains a perfect matching. We then proceed to show how to construct a perfect matching in a graph. These results are particularly interesting in the parallel context. Finally, we present the unique clique problem and show a randomized reduction from the clique problem to it.

11.1 Parallel complexity classes

The classes NC and RNC are classes of problems that can be solved by efficient parallel algorithms. Our computation model is the PRAM, where many processors can access a shared memory.

A problem is said to be in NC, if there is a PRAM (deterministic) algorithm, that uses a polynomial number of processors and works in poly-log time. A problem is said to be in RNC, if there is a randomized PRAM algorithm, that uses a polynomial number of processors, and works in poly-log time. It was shown in [Be], [Ch], [CoWi], [Cs] that problems in Linear Algebra such as matrix multiplication, computing a determinant, matrix inversion, etc., are all in the class NC.

These classes are robust, i.e., they have the property that if problem A is in (R)NC, and if problem B has a (R)NC algorithm that uses the solution of A as an atomic step, then B is also in (R)NC.

11.2 Checking if a polynomial is identically zero

Suppose we have a "black box" which computes the value of a polynomial $P(x_1, ..., x_n)$ of degree d over some field $\mathcal{F}$ for any substitution of the variables. We would like to determine whether the given polynomial is identically zero, that is - $P(\bar{x}) = 0$ for every $\bar{x} =< x_1, ..., x_n >$. This situation may occur, for example, when we have a matrix with entries that contain polynomials in the variables $x_1, ..., x_n$, and we would like to know whether or not the determinant of this matrix equals zero. Trying to compute the determinant explicitly may not be possible, since it might take exponential time even to write down the output.

A possible solution to this problem may be to choose random substitutions for $\bar{x}$, check whether $P(\bar{x}) = 0$. The following theorem (by [Sc]) asserts that if we make enough tries, with high probability we get the correct answer.

Theorem 11.2.1 *For any polynomial $P(x_1, ..., x_n)$ of degree d over a field $\mathcal{F}$ which is not identically zero, and any set $\Omega \subset \mathcal{F}$ of cardinality N, P has at most $d \times N^{n-1}$ roots in the set Ω^n.*

Proof: The proof is by induction on n:

- For $n = 1$ the proof is trivial, since a polynomial in one variable of degree d has at most d roots.
- For $n > 1$, we denote by k the highest degree of x_n that has a non-zero coefficient. Clearly $0 < k \leq d$. Let us write the polynomial P in the form:

  $$P(x_1, ..., x_{n-1}, x_n) = g_0 + g_1 x_n + g_2 {x_n}^2 + ... + g_k {x_n}^k$$

  where g_i is a polynomial in the variables $x_1, ..., x_{n-1}$ of degree at most $d - i$.

  We distinguish between two cases:

  - If $g_k(x_1, ..., x_{n-1}) = 0$: By the induction hypothesis, there are at most $(d - k) \times N^{n-2}$ vectors $\hat{x} =< x_1, ..., x_{n-1} >$ for which $g_k(\hat{x}) = 0$. Therefore, there are at most $(d - k) \times N^{n-1}$ vectors $\bar{x} =< x_1, ..., x_{n-1}, x_n >$ for which $P(\bar{x}) = 0$ and $g_k(< x_1, ..., x_{n-1} >) = 0$.
  - If $g_k(x_1, ..., x_{n-1}) \neq 0$: For every (fixed) vector $\hat{x} =< x_1, ..., x_{n-1} >$ for which $g_k(\hat{x}) \neq 0$, the polynomial $P(\hat{x}, x_n)$ is a polynomial in one variable of degree k,

and it is not identically zero. Therefore it has at most k roots. Since there are at most N^{n-1} such fixed vectors $\hat{x}$, then there are at most $k \times N^{n-1}$ vectors $\bar{x} =< x_1, ..., x_{n-1}, x_n >$ for which $g_k(< x_1, ..., x_{n-1} >) \neq 0$, but $P(\bar{x}) = 0$.

Thus, there are at most $(d-k) \times N^{n-1} + k \times N^{n-1} = d \times N^{n-1}$ vectors $\bar{x} \in \Omega^n$ for which $P(\bar{x}) = 0$. □

By the above theorem, if we pick a random vector $\bar{x} =< x_1, ..., x_{n-1}, x_n > \in \Omega^n$, then the probability that $P(\bar{x}) = 0$ (where P is not identically zero) is no more than $\frac{d \times N^{n-1}}{N^n} = \frac{d}{N}$, and if we choose a big enough set, (say $\Omega = \{-d, ..., d\}$), we get, for a random vector $\bar{x} \in \Omega^n$:

$$\Pr(P(\bar{x}) = 0 \mid P \textit{ is identically zero}) = 1$$
$$\Pr(P(\bar{x}) = 0 \mid P \textit{ is not identically zero}) \leq \frac{1}{2}$$

If we pick n such random vectors, we get the right answer with probability of at least $1 - (\frac{1}{2})^n$.

11.3 Testing similarity of matrices

Definition 11.3.1 *We say that two $n \times n$ matrices over a field $\mathcal{F}$, A and B, are similar if there exists a non-singular matrix T such that $TAT^{-1} = B$.*

Given two $n \times n$ matrices, A and B, we would like to decide whether they are similar. We can rewrite the similarity condition as $TA = BT$, where $det(T) \neq 0$. If we view the entries of T as variables, and denote by $\bar{t}$ the vector of these variables, then the latter equality is a set of n^2 linear equations, with the entries of $\bar{t}$ as variables. These equations can be expressed as $C\bar{t} = 0$, where C is an $n^2 \times n^2$ matrix.

This is a homogeneous set of equations, and we can find a basis for its solution space in NC. Let $\{\bar{t_1}, \bar{t_2}, ..., \bar{t_k}\}$ denote this basis. Any solution matrix T is of the form:

$$T = \alpha_1 T_1 + \alpha_2 T_2 + \ \dots \ + \alpha_k T_k$$

where T_i is the matrix that corresponds to the base vector $\bar{t_i}$. We are interested in a non-singular solution. Such a solution exists if and only if there is a vector $< \alpha_1, \alpha_2, \ \dots \ , \alpha_k >$ for which $det(T) \neq 0$, or, in other words, if $det(T)$ - as a polynomial in the variables $\alpha_1, \alpha_2, \ \dots \ , \alpha_k$ - is not identically zero.

Using the fact that we can evaluate $det(T)$ in NC, (for any fixed vector $< \alpha_1, \alpha_2, \ldots, \alpha_k >$), we can apply Schwartz's algorithm to check if the polynomial $det(T)$ is identically zero. This gives an RNC algorithm for testing similarity of matrices. *

11.4 Deciding if a graph contains a perfect matching

Definition 11.4.1 *Given an undirected graph $G = (V, E)$ with $2n$ vertices, a* perfect matching *in the graph is a set of n edges $\{e_1, \ldots, e_n\}$, such that no two edges share a common vertex.*

First, we present algorithms that, given a graph G, check if G contains a perfect matching. Later on, we will see an algorithm that finds a perfect matching in the graph, if one exists. The first algorithm we present decides whether a bipartite graph contains a perfect matching.

11.4.1 Deciding if a bipartite graph contains a perfect matching

Let $G = (U, V, E)$ be a bipartite graph where both U and V contain n vertices each. Consider the following $n \times n$ matrix A: The rows of A correspond to the vertices of U and the columns correspond to the vertices of V. A is defined as follows (for every $1 \leq i, j \leq n$, x_{ij} is a variable) :

$$a_{ij} = \begin{cases} x_{ij} & \text{if } (u_i, v_j) \in E \\ 0 & \text{otherwise} \end{cases}$$

The determinant of A is a polynomial of degree at most n in the variables x_{ij}

$$det(A) = \sum_{\pi} sign(\pi) a_{1,\pi(1)} a_{2,\pi(2)} \ldots a_{n,\pi(n)}$$

Clearly, every permutation π such that $a_{i,\pi(i)} \neq 0$ for all $1 \leq i \leq n$, corresponds to a perfect matching in the graph G. Furthermore, for every π, the term $x_{1,\pi(1)} x_{2,\pi(2)} \ldots x_{n,\pi(n)}$ appears at most once in $det(A)$, and can not be cancelled by any other term. Thus, the polynomial $det(A)$ is identically zero if and only if the graph G does not contain a perfect matching.

11.4.2 Deciding if a general graph contains a perfect matching

Things are more involved when we deal with a general graph. let $G = (V, E)$ be an undirected graph where $|V| = n$. We begin by arbitrarily orienting the edges of G, say

*We note that a deterministic NC algorithm for this problem was recently given by [ZG]

$v_i \to v_j$, iff $i < j$ and $(v_i, v_j) \in E$. Then, we define the $n \times n$ Tutte matrix A of G as follows: Let x_{ij} be a variable for every $1 \leq i < j \leq n$. Then,

$$a_{ij} = \begin{cases} x_{ij} & \text{if } i < j \text{ and } (v_i, v_j) \in E \\ -x_{ji} & \text{if } i > j \text{ and } (v_i, v_j) \in E \\ 0 & \text{otherwise} \end{cases}$$

We can view this matrix as if we assign x_{ij} to the entry of A that corresponds to the edge (v_i, v_j) in its "right" direction, and assign $-x_{ij}$ to the entry of A that corresponds to the same edge in the opposite direction.

Theorem 11.4.1 *The graph G has a perfect matching if and only if the determinant of its Tutte matrix is not identically zero.*

Proof: Again we can write the determinant of A as

$$det(a) = \sum_{\pi} sign(\pi) a_{1,\pi(1)} a_{2,\pi(2)} \ldots a_{n,\pi(n)}$$

but this time - not every non-zero term corresponds to a perfect matching, and a term might appear more then once in the sum.

Let π be a permutation for which $a_{1,\pi(1)} a_{2,\pi(2)} \ldots a_{n,\pi(n)} \neq 0$. We denote the sign of the permutation π by $sign(\pi)$, and by $value(\pi)$ we denote the sign of the multiplication $\alpha_{1,\pi(1)}, \alpha_{2,\pi(2)}, \ldots, \alpha_{n,\pi(n)}$.
For example, if $n = 5$ and $\pi = (2, 3, 1, 5, 4)$, then $sign(\pi) = -1$, since the pairs that are in the wrong order are $(2, 1)$, $(3, 1)$ and $(4, 5)$, so there is an odd number of such pairs. If all the edges (v_1, v_2), (v_2, v_3), (v_3, v_1), and (v_4, v_5) are in E, then $value(\pi) = +1$, since the term that corresponds to π is $x_{12} x_{23} (-x_{13}) x_{45} (-x_{45}) = +1 \times x_{12} x_{23} x_{13} x_{45}^2$.

In what follows, we view a permutation π as a combination of disjoint cycles $\{C_1, C_2, \ldots, C_k\}$. The above permutation, for example, is combined of two disjoint cycles: $\pi = \{(2, 3, 1), (4, 5)\}$. For every permutation π, we denote its inverse permutation by π^{-1}, and for every cycle C, we denote the "reversed" cycle (i.e., the same cycle in reversed order) by C^{-1}.
The following two claims state some general properties of permutations:

Claim 11.4.1 *For every permutation π: $sign(\pi) = sign(\pi^{-1})$.*

Proof: For every π: $sign(\pi) \times sign(\pi^{-1}) = sign(\pi \circ \pi^{-1}) = sign(I) = 1$ $\square$

Claim 11.4.2 *If* $\pi_1 = \{C_1, C_2, ..., C_k\}$ *and* $\pi_2 = \{C_1^{-1}, C_2, ..., C_k\}$, *then* $sign(\pi_1) = sign(\pi_2)$.

Proof: We denote by π_{C_1} the permutation that is the identity permutation for all the elements that are not in C_1, and agrees with π_1 on the elements in C_1. Clearly, $\pi_1 = \pi_{C_1} \circ \pi_{C_2} \circ ... \circ \pi_{C_k}$ and $\pi_2 = \pi_{C_1^{-1}} \circ \pi_{C_2} \circ ... \circ \pi_{C_k}$. Since $\pi_{C_1^{-1}} = (\pi_{C_1})^{-1}$, we get from claim 5.1:

$$\begin{aligned} sign(\pi_1) = \quad & sign(\pi_{C_1} \circ \pi_{C_2} \circ ... \circ \pi_{C_k}) & = \\ & sign(\pi_{C_1}) \times sign(\pi_{C_2} \circ ... \circ \pi_{C_k}) & = \\ & sign((\pi_{C_1})^{-1}) \times sign(\pi_{C_2} \circ ... \circ \pi_{C_k}) & = \\ & sign(\pi_{C_1^{-1}}) \times sign(\pi_{C_2} \circ ... \circ \pi_{C_k}) & = \\ & sign(\pi_{C_1^{-1}} \circ \pi_{C_2} \circ ... \circ \pi_{C_k}) & = sign(\pi_2) \end{aligned}$$

$\square$

The following two claims state some properties of the terms that correspond to permutations in the Tutte matrix A:

Claim 11.4.3 *If* $\pi_1 = \{C_1, C_2, ..., C_k\}$ *and* $\pi_2 = \{C_1^{-1}, C_2, ..., C_k\}$ *then*

$$value(\pi_1) = (-1)^{|C_1|} \times value(\pi_2)$$

Proof: Clearly the terms that correspond to π_1 and π_2 differ only in the variables that correspond to edges of the cycle C_1. Every edge in C_1, is either in its "right" direction in C_1, and in the "wrong" direction in C_1^{-1}, or vice-versa. Therefore, every variable that corresponds to an edge in C_1, will appear with opposite signs in the terms for π and π^{-1}, and hence the claim. $\square$

We focus now on permutation $\pi = \{C_1, ..., C_k\}$ such that at least one of the C_i's is of odd length.

Claim 11.4.4 *The total contribution to* $det(A)$ *of the permutations that contain a cycle of odd length is zero.*

Proof: For a permutation that contains a cycle of odd length, $\pi = \{C_1, ..., C_k\}$, we denote by π^r the permutation that is constructed from π by reversing the odd cycle that contains the vertex with the lowest index (among all the cycles of odd length). If we denote that cycle by C_{i_π}, then $\pi^r = \{C_1, ..., C_{i_\pi - 1}, C_{i_\pi}^{-1}, C_{i_\pi + 1}, ..., C_k\}$. Clearly since that cycle is still the odd cycle that contains the vertex with the lowest index, then $(\pi^r)^r = \pi$.

From Claims 11.4.2 and 11.4.3 we get that $sign(\pi) = sign(\pi^r)$ and $value(\pi) = -value(\pi^r)$ (since the length of C_{i_π} is odd). Therefore we get

$$sign(\pi) a_{1,\pi(1)} ... a_{n,\pi(n)} = -sign(\pi^r) a_{1,\pi^r(1)} ... a_{n,\pi^r(n)}$$

so the total contribution of π and π^r to $det(A)$ is zero. Since for every permutation π that contains odd length cycles there exists such a permutation π^r, it follows that the total contribution to $det(A)$ of these permutations is zero. $\square$

We can now complete the proof in both directions:

- If G contains a perfect matching, then the permutation that corresponds to this perfect matching is combined of $\frac{n}{2}$ cycles of length 2 each. The term that corresponds to this permutation is $\pm x_{i_1,j_1}^2 \; ... \; x_{i_{\frac{n}{2}},j_{\frac{n}{2}}}^2$ and it is not cancelled by any other term, thus $det(A) \neq 0$.

- If $det(A) \neq 0$ then, by claim 5.4, there exists a permutation π, that can be decomposed into cycles of even length, such that the term for π in $det(A)$ is not zero. Therefore all the edges that correspond to π are present in G. Thus, by numbering the edges in each cycle and choosing all the edges with odd index, we can obtain a perfect matching in G. $\square$

11.5 Computing a perfect matching in a graph

In the previous section we gave a randomized algorithm for deciding whether a graph contains a perfect matching. We proceed to show a randomized algorithm that computes a perfect matching in a graph G if it exists. The algorithm is by [MVV] and it is based on the following lemma, called "The Isolating Lemma". (We note that the first RNC algorithm for computing a perfect matching in a graph was given by [KUW1].

11.5.1 The Isolating Lemma

Lemma 11.5.1 **(The Isolating Lemma)** *Let $A = \{a_1, a_2, ..., a_m\}$ be a set of elements, and let $\mathcal{S} = \{S_1, S_2, ..., S_k\}$ be a collection of subsets of A. If we choose integer weights $w_1, w_2, ..., w_m$ to the elements of A at random from the domain $\{1, ..., n\}$, then*

$$\Pr_{w_1,...,w_m}(\textit{the minimal weight subset is not unique}) \leq \frac{m}{n}$$

where the weight of a set S is the sum of the weights of its elements.

Proof: First, consider the vector $\widehat{w_i} =< w_1, ..., w_{i-1}, w_{i+1}, ..., w_m >$, where all the weights except w_i are fixed. We will show that for every such vector, there is at most one value of w_i that violates the uniqueness property.

Claim 11.5.1 *There is a value $\alpha_{\widehat{w_i}}$ satisfying:*

- *If $w_i > \alpha_{\widehat{w_i}}$ then a_i is not a member of any minimal weight subset.*
- *If $w_i < \alpha_{\widehat{w_i}}$ then a_i is a member of every minimal weight subset.*

Proof: For the fixed weights $w_1, ..., w_{i-1}, w_{i+1}, ..., w_m$, and for every w_i, we denote the weight of the minimal subset(s) that does not contain a_i by W, and by $W'(w_i)$ we denote the weight of the minimal subset that contains a_i. Clearly $W'(w_i) = W'(0) + w_i$. Now we define $\alpha_{\widehat{w_i}} = W - W'(0)$. If $w_i < \alpha_{\widehat{w_i}}$ then we get $W > W'(w_i)$ and therefore a_i is a member in (all) the minimal weight subset(s). Similarly, if $w_i > \alpha$ then we get $W < W'(w_i)$ and therefore a_i is not a member in (any of) the minimal weight subset(s). □

We say that the element $a_i \in A$ is ambiguous with respect to $\widehat{w_i}$ if $w_i = \alpha_{\widehat{w_i}}$. It follows that for any fixed vector $\widehat{w_i}$ we get

$$\Pr_{w_i}(a_i \textit{ is ambiguous with respect to } \widehat{w_i}) = \Pr_{w_i}(w_i = \alpha_{\widehat{w_i}}) \leq \frac{1}{n}$$

and therefore

$$\Pr_{w_1,...,w_m}(\exists i \textit{ s.t. } a_i \textit{ is ambiguous with respect to } w_1, ..., w_{i-1}, w_{i+1}, ..., w_m) \leq$$

$$\sum_{i=1}^{m} \Pr_{w_i}(a_i \textit{ is ambiguous with respect to } w_1, ..., w_{i-1}, w_{i+1}, ..., w_m) \leq \sum_{i=1}^{m} \frac{1}{n} = \frac{m}{n}$$

But if no a_i is ambiguous, then every a_i is either in all the minimal subsets, or not in any minimal subset, and therefore, the minimal subset is unique. □

11.5.2 Computing a perfect matching

In order to find a perfect matching in a graph $G = (V, E)$, we choose integer weights for the edges uniformly at random from a large enough domain (say - $\{1, ..., n^2\}$). Then, we define the Tutte matrix A as follows:

$$a_{ij} = \begin{cases} 2^{w_{ij}} & \text{if } i < j \text{ and } (v_i, v_j) \in E \\ -2^{w_{ij}} & \text{if } i > j \text{ and } (v_i, v_j) \in E \\ 0 & \text{otherwise} \end{cases}$$

From the Isolating Lemma we get that if G contains a perfect matching, then the probability that the minimal weight perfect matching is not unique is small (not more than $\frac{|E|}{n^2} \leq \frac{1}{2}$).

Claim 11.5.2 *If the minimal weight perfect matching in G is unique then:*

- *$det(A) \neq 0$.*
- *The highest power of two that divides $det(A)$ is 2^{2W}, where W is the minimal weight of a perfect matching in G.*

Proof: As before, we can ignore the permutations that contain cycles of odd length, since their total contribution to $det(A)$ is zero. Let us denote the minimal weight perfect matching by M. M is combined of $\frac{n}{2}$ cycles of length 2 each. For every edge (v_i, v_j) in this matching, the factor $2^{w_{ij}}$ appears twice in the term of the permutation (once as a positive factor and once as a negative one). Thus, the total contribution of that term is $\pm 2^{2W}$. Any other permutation π (with no odd cycle), for which all the edges are present in G, must belong to one of the following two categories:

- π may be a combination of cycles of length 2. In this case, π corresponds to another perfect matching M' of weight W'. The term of π will have a weight of $\pm 2^{2W'}$. Since M is the only minimal matching, we know that $W < W'$, thus 2^{2W} divides $2^{2W'}$ and $\frac{2^{2W'}}{2^{2W}}$ is even.
- π may be a combination of cycles of even length, where the length of at least one of them is grater than 2. In this case, we can view π as a combination of two different perfect matchings: M_1 (of weight W_1) that is combined of the odd-indexed edges in every cycle, and M_2 (of weight W_2) that is combined of the even edges in every cycle.

This term will have a contribution of $2^{W_1+W_2}$. Since $M_1 \neq M_2$, at least one of them is not the minimal weight perfect matching M, thus $2W < W_1 + W_2$ and therefore 2^{2W} divides $2^{W_1+W_2}$ and $\frac{2^{W_1+W_2}}{2^{2W}}$ is even.

From the above discussion we get that for every permutation π that does not contain cycles of odd length the following holds:

$$\frac{1}{2^{2W}} a_{1,\pi(1)} a_{2,\pi(2)} \ldots a_{n,\pi(n)} = \begin{cases} 1 & \text{if } \pi \text{ is the permutation that corresponds to } M \\ 0 \ (mod\ 2) & \text{otherwise} \end{cases}$$

Thus, if we write $det(A)$ in the form:

$$det(a) = 2^{2W} \times \sum_{\pi} \frac{1}{2^{2W}} sign(\pi) a_{1,\pi(1)} a_{2,\pi(2)} \ldots a_{n,\pi(n)}$$

then the sum will be odd and hence the claim. □

11.5.2.1 Constructing the minimal weight perfect matching

The above claim yields an NC algorithm to find 2^{2W} (once we fix the weights w_{ij}): Simply calculate $det(A)$ and find the highest power of 2 that divides it. We now show how to find the minimal weight perfect matching M in G:

For every edge $(v_i, v_j) \in E$, let A_{ij} denote the *minor* obtained from A by removing the i^{th} row and the j^{th} column, and let us define:

$$m_{ij} = \sum_{\pi \ s.t.\ \pi(i)=j} sign(\pi) a_{1,\pi(1)} a_{2,\pi(2)} \ldots a_{n,\pi(n)}$$

Claim 11.5.3 *For every edge $(v_i, v_j) \in E$ it holds that*

$$2^{w_{ij}} det(A_{ij}) = \pm m_{ij}$$

Proof: Let us number the rows of A_{ij} as $1, 2, \ldots, i-1, i+1, \ldots n$, and let us number the columns of A_{ij} as $1, 2, \ldots, j-1, j+1, \ldots n$. Clearly, from any permutation π of A_{ij} (in its new numbering) we can construct a permutation π' of A by adding the additional rule $\pi(i) = j$. The term of π' in $det(A)$ is identical to the term of π in $det(A_{ij})$, except that it has the additional factor $a_{ij} = \pm 2^{w_{ij}}$. Since every permutation π' that satisfies $\pi(i) = j$ can be constructed this way, the claim follows. □

Claim 11.5.4 *For every edge $(v_i, v_j) \in E$ the following holds:*

- *The total contribution to m_{ij} of permutations that contain a cycle of odd length is zero.*
- *If $(v_i, v_j) \in M$ then $\frac{m_{ij}}{2^{2W}}$ is an odd number.*
- *If $(v_i, v_j) \notin M$ then $\frac{m_{ij}}{2^{2W}}$ is an even number.*

Proof:

- As before, for every such permutation π we can assign a permutation π^r satisfying $(\pi^r)^r = \pi$ and $sign(\pi)a_{1,\pi(1)}...a_{n,\pi(n)} = -sign(\pi^r)a_{1,\pi^r(1)}...a_{n,\pi^r(n)}$. The only difficulty is that we might not be able to reverse the first odd cycle: If that cycle contains the edge (v_i, v_j) then reversing it will generate a permutation π' for which $\pi'(i) \neq j$. However, we can easily solve this difficulty. Since G contains an even number of vertices (or else - G can not contain a perfect matching), then every permutation that contains a cycle of odd length, contains at least two such cycles. If the first odd cycle contains the edge (v_i, v_j) then the second one does not contain it, since the cycles are disjoint. Thus we define π^r as the permutation that is constructed from π by reversing

  - the first odd cycle, if it does not contain the edge (v_i, v_j).
  - the second odd cycle, if the first contains the edge (v_i, v_j).

  It can be easily shown that for every permutation π that contains a cycles of odd length and that satisfies $\pi(i) = j$ it holds that

  - $\pi^r(i) = j$
  - $(\pi^r)^r = \pi$
  - $sign(\pi)a_{1,\pi(1)}...a_{n,\pi(n)} = -sign(\pi^r)a_{1,\pi^r(1)}...a_{n,\pi^r(n)}$

  and the claim follows.

- If $(v_i, v_j) \in M$ then the sum m_{ij} contains the term that corresponds to M and the weight of this term is 2^{2W}. As before, all the other non-zero terms corresponds to "heavier" matchings, and therefore, 2^{2W} is the highest power of two that divides m_{ij}.
- If $(v_i, v_j) \notin M$ then there can be two possibilities:

  - If $m_{ij} = 0$, the claim follows.

– If $m_{ij} \neq 0$ then there are perfect matchings in G that contains the edge (v_i, v_j). Let W' denote the minimal weight of a perfect matching that contains that edge. Then, $W < W'$ (since M is the only perfect matching of weight W). It is easy to show, using the same arguments as before, that $2^{2W'}$ is the highest power of two that divides m_{ij}, and since $W < W'$, $\frac{m_{ij}}{2^{2W}}$ is an even number. □

The algorithm to find a perfect matching in G (if one exists) is the following:

- If the number of vertices in G is odd - then G doesn't contain a perfect matching.
- Fix integer weights to the edges of G at random from the domain $\{1, ..., N\}$.
- Define the Tutte matrix A according to these weights, calculate $det(A)$, and find the highest power of two that divides it. Let us denote this power of two by x.
- For every edge $e = (v_i, v_j) \in E$, if $\frac{2^{w_{ij}} det(A_{ij})}{2^x}$ is odd then mark e as belonging to M. Otherwise, mark e as not belonging to M.
- If the edges belonging to M do not form a perfect matching in G, then G does not contain a perfect matching with high probability.

From the above claims we get that the algorithm can fail only if the minimal weight perfect matching is not unique. By the Isolating Lemma, this happens with probability of at most $\frac{|E|}{N}$.

Open research problem: Find a deterministic NC algorithm for computing a perfect matching in a graph.

11.6 The exact matching problem

Definition 11.6.1 *The* exact matching problem *defined in [PY], is the following:*

- *Input : An undirected graph $G = (V, E)$, a coloring of the edges of G in red and blue, and an integer k.*
- *Problem : Find a perfect matching in G that contains exactly k red edges.*

To solve the exact matching problem, we can pick weights to the edges of G uniformly at random from the domain $\{1, 2, ..., N\}$, and then define the Tutte matrix A as follows:

$$a_{ij} = \begin{cases} 2^{w_{ij}} & \text{if } i < j \text{ and } (v_i, v_j) \text{ is a blue edge} \\ -2^{w_{ij}} & \text{if } i > j \text{ and } (v_i, v_j) \text{ is a blue edge} \\ 2^{w_{ij}}y & \text{if } i < j \text{ and } (v_i, v_j) \text{ is a red edge} \\ -2^{w_{ij}}y & \text{if } i > j \text{ and } (v_i, v_j) \text{ is a red edge} \\ 0 & \text{otherwise} \end{cases}$$

where y is a variable.

The determinant of A in this case is not a number, but a polynomial of degree at most n in the variable y. Since this is a determinant in one variable, we can efficiently compute it in NC. Note that the term in the determinant corresponding to a perfect matching that contains exactly k red edges, contains the factor y^{2k}, and these are the only terms that contain this factor. Therefore, we can apply the algorithm of Section 11.5 for finding a perfect matching in a graph to the coefficient of y^{2k}. As before, the algorithm can fail only if the minimum weight perfect matching (among all perfect matchings with exactly k red edges) is not unique, and that happens with probability of at most $\frac{|E|}{N}$.

Open research problem: Find a deterministic algorithm for the exact matching problem. Notice that in contrast to the perfect matching problem, even a sequential deterministic algorithm is not known for this problem!

11.7 The unique clique problem

In this section we see another application of the Isolating Lemma. We define the unique clique problem, which is a variant of the clique problem, and show a randomized reduction from the unique clique problem to the clique problem.

11.7.1 Promise problems

A promise problem is a problem where you are given, beside your input, some "promise" associated with the input. We say that an algorithm solves a promise problem, if this algorithm gives the right answer on every input that satisfies the promised condition. The algorithm may fail, however, on inputs that does not satisfy the promised condition. Clearly, the promise is of any significance only if the promised condition cannot be verified (using the same resources as the algorithm).

11.7.2 The unique clique problem

Definition 11.7.1 *The clique problem is defined as follows:*

- *Input: An undirected graph $G = (V, E)$ and an integer k.*
- *Question: Does G contain a clique of size k?*

Definition 11.7.2 *The unique clique problem is a promise problem that is defined as follows:*

- *Input: An undirected graph $G = (V, E)$ and an integer k.*
- *Promise: G contains AT MOST ONE clique of size k.*
- *Question: Does G contain a clique of size k?*

Claim 11.7.1 *If the unique clique problem has a BPP/RP/RNC algorithm that solves it, then every problem in NP has such an algorithm.*

(BPP is the class of all problems that have polynomial-time randomized algorithms. RP is the class of all problems that have polynomial-time randomized algorithms with only one-sided error).

Proof: We prove the claim by presenting an efficient randomized one-sided error reduction from the unique clique problem to the clique problem which is known to be NP-complete. This reduction is defined as follows:

- Input: An undirected graph $G = (V, E)$ ($|V| = n$) and an integer k.
- Output: An undirected graph $G' = (V', E')$ (where $|V'| = Poly(n, k)$) such that:
  - If the maximum clique in G is strictly smaller than k, then the maximum clique in G' is strictly smaller than $2nk^2$.
  - If the maximum clique in G is of size k, then with high probability, there is an integer r, $2nk^2 \leq r \leq 2nk(k+1)$, such that G' contains exactly one clique of size r.

Such a reduction will prove the claim. On an input G to the clique problem, we can apply the reduction for every possible k (up to n) and generate the graph G'. Then, we can run the unique clique algorithm on the graph G' where r is either chosen uniformly at random from the range $[2nk^2, 2nk(k+1)]$, or we can check all the integers in this range. Assuming the reduction can be carried out in parallel, since $k \leq n$, the claim follows.

Given a graph G and an integer k, we set weights to the vertices of G uniformly at random from the domain $\{1, 2, ..., 2n\}$. Then, we define G' as follows: We replace every vertex $v \in V$ by a clique C_v of size $2nk + w(v)$, and for every edge $(u, v) \in E$ we connect all the vertices in C_u to all the vertices in C_v. Formally, we define:

$$V' = \bigcup_{v \in V} \{v_i : 1 \leq i \leq 2nk + w(v)\}$$
$$E' = \{(v_i, v_j) : v \in V\} \cup \{(u_i, v_j) : (u, v) \in E\}$$

Note that if $C = \{v_1, ..., v_t\} \subseteq V$ is a clique in G, then $C' = C_{v_1} \cup C_{v_2} \cup ... \cup C_{v_t} \subseteq V'$ is a clique in G' of cardinality $2nkt + w(v_1) + w(v_2) + ... + w(v_t)$. Note also that every (maximal) clique in G' corresponds to a clique (or a single vertex) in G.

Claim 11.7.2 *If the maximum clique in G is strictly smaller than k, than the maximum clique in G' is strictly smaller than $2nk^2$.*

Proof: The size of any clique $C' \subseteq V'$ that corresponds to a clique $C = \{v_1, ..., v_{k-1}\} \subseteq V$ is at most:

$$2nk(k-1) + w(v_1) + w(v_2) + ... + w(v_{k-1}) \leq$$
$$2nk(k-1) + 2n(k-1) = 2n(k+1)(k-1) < 2nk^2$$

□

Claim 11.7.3 *If the maximum clique in G is of size k, then with high probability, there is an integer r, $2nk^2 \leq r \leq 2nk(k+1)$, such that G' contains exactly one clique of size r.*

Proof: From the Isolating Lemma we get that, with high probability, the maximum weight clique, (among all cliques of size k in G) is unique. Let us denote this clique by C, let W denote its weight, and let C' denote the corresponding clique in G'. C' is a clique of cardinality $2nk^2 + W$. Note that $2nk^2 < 2nk^2 + W < 2nk(k+1)$. For any clique $\bar{C} \subseteq V$ of size k, with weight $\bar{W}$, The corresponding clique in G' is of cardinality $2nk^2 + \bar{W} < 2nk^2 + W$. By claim 6.2, for every clique in G of size $< k$, the corresponding clique in G' is smaller than $2nk^2 < 2nk^2 + W$. Since there are no cliques in G of size $> k$, it follows that G does not contain a clique larger then C'. Therefore, C' is the maximum clique in G' and it is unique. This concludes our proof. □

Chapter 12

Linear Programming

12.1 Introduction

In these lectures we will see some examples of probabilistic algorithms used in Linear programming. The summary is organized as follows - In sections 12.2 and 12.2.1 we give a general description of linear programming ([Ka]). In Section 12.2.3 the Duality Theorem is presented. Then, we demonstrate its use in proving König's Theorem. A few deterministic algorithms are briefly mentioned in section 12.2.4. Finally, In section 12.3 we demonstrate two randomized algorithms ([Se], [Cl]) for the fixed dimension case.

12.2 Linear programming

Linear programming is the process of minimizing a linear *objective function* subject to a finite number of linear equality and inequality constraints. Let $x, c \in \mathcal{R}^n$, let A be a $k \times n$ matrix, let $b \in \mathcal{R}^k$, let A' is a $l \times n$ matrix and let $b' \in \mathcal{R}^\ell$. The following is a general linear program with n variables, k equality constraints, and ℓ inequality constraints.

$$\begin{aligned} \text{minimize} \quad & c^T x \\ \text{subject to} \quad & Ax = b \\ & A'x \geq b' \end{aligned}$$

$$x_1 \geq 0 \quad x_2 \geq 0 \ \cdots \ x_r \geq 0 \quad x_{r+1} \gtrless 0 \ \cdots \ x_n \gtrless 0$$

The notation $x_j \gtrless 0$ means that variable x_j is not sign constrained. Either k or ℓ can be 0.

Consider two special classes of linear programs.

The Standard form :

$$\begin{aligned} \text{minimize} \quad & c^T x \\ \text{subject to} \quad & Ax = b \\ & x \geq 0 \end{aligned}$$

In which every variable is sign-constrained and only equality constraints are allowed.

The Canonical form :

$$\begin{aligned} \text{minimize} \quad & c^T x \\ \text{subject to} \quad & Ax \geq b \\ & x \geq 0 \end{aligned}$$

a form which allows only inequality constraints and in which all variables are sign-constrained.

All three forms are equivalent in that one can easily be converted to any other. Clearly, a linear program in standard or canonical form is already in general form, so to prove equivalence it suffices to show that general-form linear programs can be converted to standard and canonical form. To this end we should describe how to eliminate equality constraints, how to eliminate inequality constraints, and how to eliminate unsign-constrained variables.

eliminate inequality - Let $d_i^T x \geq r_i$ be an inequality constraint. Create a new "surplus variable" s_i and replace the inequality $d_i^T x \geq r_i$ by $d_i^T x - s_i = r_i$ and $s_i \geq 0$

eliminate equality - Let $d_i^T x = r_i$ be an equality constraint. Replace the equality constraint by a pair of inequality constraints $d_i^T x \geq r_i, \;\; -d_i^T x \geq -r_i$

eliminate unsign-constraint - If $x_j \gtrless 0$, create two new variables x_j^+ and x_j^- and replace x_j everywhere by $x^+ - x^-$, and add the sign-constraints $x^+ \geq 0, \;\; x^- \geq 0$

It is easy to see that the resulting problems are equivalent to the original problem.

12.2.1 Geometrical interpretation

We start with a few geometric definitions.

Definition 12.2.1 *A* hyperplane *in* $\mathcal{R}^n$ *is*

$$H = \{x \mid a_1x_1 + \cdots + a_nx_n = b\},$$

where not all $a_i = 0$. *A* halfspace *is the set* $\{x \mid a_1x_1 + \cdots a_dx_d \geq b\}$ *where not all* $a_i = 0$.

We say that a set $T \subseteq \mathcal{R}^n$ is *bounded* if there is a real r such that $\|x\| \leq r$ for all $x \in T$.

Definition 12.2.2 *A* polyhedron *is the intersection of finitely many halfspaces. A bounded nonempty polyhedron is called a* polytope.

Example 12.2.1: The set $\{x \mid Ax \geq b,\ x \geq 0\}$ is the intersection of finitely many halfspaces, as is $\{x \mid Ax = b,\ x \geq 0\}$. If they are nonempty and bounded, they are called polytopes.

We say that a sequence of equality or inequality constraints are *linearly independent* if the sequence of coefficients' vectors are linearly independent. We ignore the right-hand sides and the question whether the constraints are equalities or inequalities.

Definition 12.2.3 *If* $x, y, p \in \mathcal{R}^n$ and $p = \lambda x + (1 - \lambda)y,\ 0 \leq \lambda \leq 1$ *then* p *is the* convex combination *of* x *and* y. *More generally,* p *is the* convex combination *of points* $x_1, x_2, \ldots, x_r$ *if*

$$p = \sum_{i=1}^{r} \lambda_i x_i, \quad \lambda_i \geq 0, \quad \text{and} \quad \sum_{i=1}^{r} \lambda_i = 1.$$

Point p *is the* strict convex combination *of points* $x_1, x_2, \ldots, x_r$ *if* p *is the convex combination of* $x_1, x_2, \ldots, x_r$ *yet* $p \neq x_i$ *for all* i.

Definition 12.2.4 $S \subseteq \mathcal{R}^n$ *is* convex *if whenever* $x, y \in S$, *then every convex combination of* x *and* y *is in* S. *That is, the line segment connecting* x *and* y *lies in* S.

It is easy to see that the intersection of arbitrary many convex sets is convex.

12.2.2 Basic Solutions

Definition 12.2.5 *If* P *is a polyhedron,* $v \in P$ *is an* extreme point *or a* vertex *if* v *cannot be written as the strict combination of points* $x, y \in P$.

Definition 12.2.6 *If* A *is an* $m \times n$ *matrix of rank* m, *any* m *linearly independent columns are a* basis. *(they form a basis of* A*'s column space,* $\mathcal{R}^m$*).*

Definition 12.2.7 *Consider the linear program*

$$\begin{aligned} \text{minimize} \quad & c^T x \\ \text{subject to} \quad & Ax = b \\ & x \geq 0 \end{aligned}$$

where A is of rank m and has dimension $m \times n$. If columns $j_1 < j_2 < \cdots < j_m$ are a basis, the corresponding basic solution *x is defined as follows and ignores c. Let B be the nonsingular matrix consisting of columns $j_1, j_2, \ldots, j_m$ of A. Any variable x_ℓ where $\ell \notin \{j_1, j_2, \ldots, j_m\}$ is set to 0 and is called* nonbasic. *For $k = 1, \ldots, m$, x_{j_k} is set to the k^{th} component of $B^{-1}b$ and is known as* basic; *the basic variables may or may not be 0. (The basic solution x in the unique vector satisfying $Ax = b$ subject to the condition that $x_\ell = 0$ for $\ell \notin \{j_1, \ldots, j_m\}$.) If x satisfies the sign constraints $x \geq 0$, x is a* basic feasible solution *or* bfs.

If $m \times n$ matrix A has rank m, then $Ax = b$ always has a basic solution. If the feasible region is nonempty, we see that it must have a bfs as well.

Notation 12.2.1 *Let us use LPS to denote*

$$\begin{aligned} \text{minimize} \quad & c^T x \\ \text{subject to} \quad & Ax = b \\ & x \geq 0 \end{aligned}$$

where A is an $m \times n$ matrix of rank m.

We use F to denote the feasible set $\{x \mid Ax = b,\ x \geq 0\}$.

Theorem 12.2.1 *In LPS, suppose that $w \in \mathcal{R}^n$. Then w is a vertex of the feasible set F if and only if w is a basic feasible solution of LPS.*

Corollary 12.2.1 *There are only finitely many vertices.*

Proof: Each is a bfs which is determined by an m-element subset of the columns. The number of bfs's is therefore at most $\binom{n}{m}$. □

Say a minimizing problem is *unbounded* if for each real B, there is a feasible point of cost less then B.

Theorem 12.2.2 *In LPS, suppose $p \in F$. Then either LPS is unbounded, or there is a vertex v of F satisfying $c^T v \leq c^T p$.*

Corollary 12.2.2 *If in LPS $F \neq \phi$* and $c^T x \geq B$ for all $x \in F$, *then there is an optimal vertex. Provided that the cost is bounded from below, linear programming is a finite problem!*
Proof sketch *Let $p_1 \in F$. Find a vertex v_1 such that $c^T v_1 \leq c^T p_1$. (Theorem 12.2.2 guarantees v_1's existence.) If v_1 is optimal we are done. Otherwise, there is a feasible point p_2 such that $c^T p_2 < c^T v_1$, and a vertex v_2 such that $c^T v_2 \leq c^T p_2 < c^T v_1$. If v_2 is optimal we are done. In this way we construct a sequence of vertices $v_1, v_2, \ldots$ each of cost strictly less then the previous one's. No vertex can appear twice on this list, and since there are only finitely many vertices, the process must terminate with an optimal bfs.* □

12.2.3 Duality

The original linear programming problem of minimizing an objective function subject to linear constraints is called the *primal linear programming problem.*

Associated with this is the dual problem, which has as many variables as the primal problem has constraints, and as many constraints as there are variables in the primal problem. Here is how we build the dual in general.

	PRIMAL	DUAL
row i	$\sum_j a_{ij} x_j = b_i$	$y_i \gtrless 0$
row i	$\sum_j a_{ij} x_j \geq b_i$	$y_i \geq 0$
var j	$x_j \gtrless 0$	$\sum_{i=1}^m y_i a_{ij} = c_j$
var j	$x_j \geq 0$	$\sum_{i=1}^m y_i a_{ij} \leq c_j$
objective function	min $c^T x$	max $y^T b$

To get the dual put the coefficients of the objective function on the right hand side of the dual. Read down the primal's columns and use the entries in one column to write down one dual constraint. A dual variable y_i is sign constrained ($y_i \geq 0$) if and only if the corresponding primal constraint is an inequality. A dual constraint is an equality if and only if the corresponding primal variable is sign constrained ($x_j \geq 0$). Remember, inequality in one problem corresponds to inequality in the other. Equality in one corresponds to $\gtrless$ in the other. If the primal problem is a minimization problem, then the dual problem is a maximization problem.

Example 12.2.2:

$$\begin{array}{llrrrrl}
\text{PRIMAL} & \min & 3x_1 & -6x_2 & -x_3 & +2x_4 & \\
 & & 2x_1 & -16x_2 & +12x_3 & -0x_4 & \geq 12 \\
 & & -5x+1 & -5x_2 & +0x_3 & +13x_4 & \geq -23 \\
 & & -4x_1 & -3x_2 & -x_3 & -7x_4 & = 28
\end{array}$$

$$x_1 \geq 0 \;\; x_2 \gtrless 0 \;\; x_3 \gtrless 0 \;\; x_4 \geq 0$$

$$\begin{array}{llrrrl}
\text{DUAL} & \max & 12y_1 & -23y_2 & +28y_3 & \\
 & & 2y_1 & -5y_2 & -4y_3 & \leq 3 \\
 & & -16y_1 & -5y_2 & -3y_3 & = -6 \\
 & & 12y_1 & +0y_2 & -y_3 & = -1 \\
 & & 0y_1 & +13y_2 & -7y_3 & \leq 2
\end{array}$$

$$y_1 \geq 0 \;\; y_2 \geq 0 \;\; y_3 \gtrless 0$$

Theorem 12.2.3 *If w, u is a primal/dual feasible pair, $c^T w \geq u^T b$.*

We now state the extremely important **Duality Theorem**.

Theorem 12.2.4 *If a linear program has an optimal solution, so does its dual, and their optimal costs are identical.*

12.2.3.1 A combinatorial application of duality

The major applications of linear programming are naturally in the field of optimization. Many problems in combinatorial optimization, e.g., max flow, minimal cost flow, multi-commodity flow, are instances of linear programming. In this section we present an application in combinatorics, a realm where it is less obvious that linear programming can be of use. We provide an alternative proof to König's Theorem via duality.

Given a graph G, We denote by $\tau(G)$ the size of a minimal vertex cover in G, and by $\nu(G)$ the size of a maximum matching in G.

Theorem 12.2.5 (König) *If $G = (V, U, E)$ is a bipartite graph, then $\tau(G) = \nu(G)$.*

Proof: We can formulate the maximum matching problem as an integer linear program. We create a variable x_{ij} for each edge $(i, j) \in E$, whose value is either 0 or 1 : if the solution assigns $x_{ij} = 1$, then edge (i, j) is in the matching. Otherwise, if $x_{ij} = 0$, then edge (i, j) is not in the matching.

In a valid matching there is at most one edge incident to each vertex. We therefore write a constraint for each vertex, limiting the number of edges incident to that vertex to at most one. Hence, for each node i we have the constraint $\sum_j x_{ij} \leq 1$.

Since our goal is to maximize the number of edges chosen, we want to maximize $\sum_{i,j} x_{ij}$, or minimize $-\sum_{i,j} x_{ij}$. This representation of the maximum matching problem is an integer linear program since x_i is either 0 or 1.

The following is a linear programming relaxation of the maximum matching problem, i.e., each variable x_{ij} can take a value between 0 and 1. (The constraint $x_{ij} \leq 1$ is implicit).

$$\begin{gathered}\text{minimize} \quad -\sum_{(i,j)\in E} x_{ij} \\ \text{s.t. } \forall i \quad -\sum_j x_{ij} \geq -1 \\ \forall (i,j) \in E \;\; x_{ij} \geq 0\end{gathered}$$

The dual problem has a variable y_i for each constraint in the primal problem, and a constraint for each variable x_{ij} in the primal problem.

Here is the dual problem :

$$\begin{gathered}\text{maximize} \quad -\sum_i y_i \\ \text{s.t. } \forall (i,j) \in E \quad -y_i - y_j \leq -1 \\ \forall i \;\; y_i \geq 0\end{gathered}$$

We can now see that the dual problem is the linear program formulation of the minimal vertex cover problem. For each vertex i in G, there is a variable $y_i \geq 0$. In a valid vertex cover there is at least one vertex incident to each edge. We, therefore, have a constraint for each edge, requiring that the number of vertices incident to an edge is at least one, i.e., $y_i + y_j \leq 1$. These constraints imply, that each y_i is between 0 and 1.

Since the goal in vertex cover is to minimize the number of vertices chosen, the objective function can be written as maximizing $\sum_i y_i$.

By the Duality Theorem, if the primal problem has an optimal solution so does its dual program, and their optima are equal. To complete the proof, we need to show that there exist an optimal solution which is also integral. That is, a solution in which every x_i and every y_i is either 0 or 1 (and not 0.5, for example).

A matrix is called *Totally Unimodular* (TUM) if the determinant of each submatrix is either 1, -1 or 0. An important property of TUM matrices is that the vertices of the feasible polytope defined by a TUM matrix are always integral. It can be shown that the constraint matrix in our problem is TUM, and by virtue of this we get integrality for free. $\square$

12.2.4 Deterministic algorithms

Designed in 1947 by G. Dantzig, the *Simplex Algorithm* could hardly be simpler. We already know that linear programming is a finite problem. When the objective function is bounded below, we need only scan all of the (exponentially many) vertices of the polytope in order to find an optimal point. The simplex algorithm is nothing more than an orderly way of scanning the vertices.

The algorithm starts from an arbitrary vertex v of the feasible region and finds a cheaper neighboring vertex, a neighboring vertex being a different bfs whose corresponding basis differs from that of v in exactly one column. If no neighboring vertex has cheaper cost, the current vertex can be proven to be globally optimal.

It can also be proven that in the worst case, the running time of the Simplex Algorithm is exponential. In practice, however, this algorithm turns out to be very fast.

Let d be the number of variables, let m be the number of constraints, and let ℓ be the length of the input. An algorithm for linear programming that terminates after a number of steps which is a polynomial in m and d is called a *strongly polynomial algorithm.* An algorithm that terminates after a number of steps which is polynomial in m, d and ℓ is called a *weakly polynomial algorithm.*

It is an open question whether there is a strongly polynomial algorithm for linear programming. The first weakly polynomial algorithm for linear programming, the *Ellipsoid Algorithm* was constructed by L.G. Khachiyan in 1979. This algorithm generates a sequence of points *outside* the feasible region of the linear program. Such an algorithm is an *exterior-point method.* The simplex algorithm, on the other hand, generates vertices of the feasible region. Unlike both, Karmarkar's Algorithm, designed In 1984, is an *interior-point method.* It generates a sequence of points inside the feasible region whose cost approaches the optimal cost. At the end, it jumps to a vertex of no greater cost, which is then optimal.

12.3 Linear programming in small dimension

In some applications of linear programming, the number of variables, d, is very small compared to the number of constraints, m (i.e., $d \ll m$). We will present two randomized algorithms that have expected running time which is linear or almost linear in m.

12.3.1 Seidel's algorithm

The first algorithm is due to Raimund Seidel [Se]. In this algorithm, the dependence of the time bound on d is of the form $d!$. The main virtue of this algorithm lies in the simplicity of the algorithm and its analysis.

The input to the algorithm is a set $\mathcal{H}$ of m constraints (halfspaces in $\mathcal{R}^d$), and an objective function $c \in \mathcal{R}^d$. The algorithm outputs an optimum vertex v of the polyhedron $P_{\mathcal{H}}$ formed by the intersection of the half-spaces in $\mathcal{H}$ (with respect to c).

The Algorithm

1. Remove, uniformly at random, a half-space H from the set $\mathcal{H}$ to obtain the set $\mathcal{H}'$.
2. Recursively compute the optimum vertex v' of $P_{\mathcal{H}'}$ (with respect to c).
3. From v' compute v as follows :
   - If v' is contained in the half-space H, then clearly $v = v'$, and nothing needs to be done.
   - Otherwise, v must be contained in the bounding hyperplane h of the half-space H. As a matter of fact, if $\bar{c}$ is the projection of c into h and $\bar{\mathcal{H}}' = \{G \cup h \mid G \in \mathcal{H}'\}$, then v is the optimum vertex for the $(d-1)-$ dimensional linear program specified by the half-spaces $\bar{\mathcal{H}}'$ and the objective function $\bar{c}$. Thus, v can be determined by recursively solving a $d-1$ dimension problem with $m-1$ constraints. (A one-dimensional problem can be solved straightforwardly in time proportional to the number of its constraints.)

For the sake of analysis let us assume that our linear programming problem and all subproblems encountered are well behaved in the sense that the optimum vertex is unique, and that it is the intersection of the bounding hyperplanes of exactly d of the given half-spaces.

We claim that the algorithm has an expected running time of $O(d!m)$. The proof, by induction on d, proceeds as follows:

- In case $d = 1$ the problem can be solved trivially in $O(m)$ time.
- For $d > 1$ it suffices to show that the expected time necessary to obtain v from v' is $O(d!)$. The interesting and expensive case happens when v and v' are different. But note that these vertices can only be different if one of the d half-spaces whose bounding hyperplane define v is H. Since H was chosen from the m half-spaces in $\mathcal{H}$ uniformly at random, it follows that v is different from v' with probability d/m. By inductive assumption the expected cost of solving the $d-1$-dimensional linear program problem with $m-1$ constraints is $O((d-1)!(m-1))$. Thus the expected cost of obtaining v from v' is $(d/m) \cdot O((d-1)!(m-1))$, which is $O(d!)$ as claimed.

12.3.2 Clarkson's algorithm

The second algorithm is due to Clarkson. This algorithm requires an expected time of $O(d^2m) + (\log_d m)^{\log d} \cdot O(d)^{d/2+O(1)}$. The leading term in the dependence on m is $O(d^2)$, a considerable improvement in d (compared with the $d!$ dependence in Seidel's algorithm). The second term arises from the solution of $O(\log_d m)^{\log d}$ “”small” linear programs with $O(d^2)$ constraints and d variables. The solution of these linear programming problems with the simplex algorithm requires $O(d)^{d/2+O(1)}$ time.

The algorithm is based on the following fact - the optimum point v, is determined by exactly d constraints from $\mathcal{H}$. That is, there is a set $\mathcal{H}^* \subset \mathcal{H}$ of size d, such that the optimum for $\mathcal{H}^*$ is the same as for $\mathcal{H}$. The constraints $\mathcal{H} \setminus \mathcal{H}^*$ are redundant, in the sense that their deletion from $\mathcal{H}$ does not affect the optimum.

The main idea of the algorithm is to throw away redundant constraints quickly. The algorithm builds a set $V^* \subset \mathcal{H}$ over several phases. In each phase, a set $V \subset \mathcal{H} \setminus V^*$ is added to V^*. The set V has two important properties: its size is no more then $\sqrt{m}$, and it contains a constraint from $\mathcal{H}^*$. After d phases, $\mathcal{H}^* \subset V^*$, and also V^* has $O(d\sqrt{m})$ elements. That is, the algorithm quickly throws away the large set of redundant constraints $\mathcal{H} \setminus V^*$. The algorithm proceeds recursively with V^*, and the recursion terminates for “small” sets of constraints. For these constraints, the optimum can be found using the simplex algorithm.

The Algorithm

Function LP($\mathcal{H}$: set of half-spaces)
return v : optimum vertex
$V^* \leftarrow \phi$; $C_d \leftarrow 9d^2$;
if $m \le C_d$ then return Simplex($\mathcal{H}$)
repeat
  choose $R \subset \mathcal{H}$ at random, s.t. $\mid R \mid = r = 2d\sqrt{m}\log m$;
  $v \leftarrow$ LP($\mathcal{R} \cup V^*$);
  $V \leftarrow \{H \in \mathcal{H} \mid v$ violates $H\}$;
  if $\mid V \mid \le \sqrt{m}$ then $V^* \leftarrow V^* \cup V$;
until $V = \phi$;
return v ;
end ;

We say that an execution of the loop body in LP is *successful* if $\mid V \mid \le \sqrt{m}$.

Lemma 12.3.1 *The number of successful executions is at most d.*

Proof: If $V \neq \phi$ then it must contain at least one constraint from $\mathcal{H}^*$. In every successful execution we add V to V^*. Thus, since the size of $\mathcal{H}^*$ is at most d, the proof follows.

Lemma 12.3.2 *An execution of the loop is successful with probability at least $1 - \frac{1}{m^d}$.*

Proof: Let v be a vertex that is the optimum of some d constraints from $\mathcal{H}$.

Let BAD_v be the set of constraints violated by v'. I.e., $BAD_v = \{H \in \mathcal{H} \mid v'$ violates $H\}$. Let $b(v)$ denote the cardinality of the set BAD_v. Each time a constraint H from $\mathcal{H}$ is chosen at random and placed in R, the probability that $H \in BAD_v$ is $b(v)/m$. Thus, the probability that $H \notin \mathcal{H}^*$ is $1 - b(v)/m$. If $b(v) > \sqrt{m}$, then the probability that when we chose the constraints in R, we did not hit any constraint in BAD_v, is therefore, bounded above by

$$(1 - \frac{b(v)}{m})^r \le (1 - \frac{\sqrt{m}}{m})^r = (1 - \frac{1}{\sqrt{m}})^{2d\sqrt{m}\log m} =$$

$$= ((1 - \frac{1}{\sqrt{m}})^{\sqrt{m}})^{2d\log m} = e^{-2d\log m} = \frac{1}{m^{2d}}$$

Since there are at most m^d vertices, the probability of choosing a vertex v for which $b(v) > \sqrt{m}$ (i.e. the execution will not be a successful one), is bounded above by $\frac{1}{m^d}$. □

Corollary 12.3.1 *The expected number of loop executions is $O(d)$.*

Proof: Every execution is successful with probability greater than 1/2, therefore the expected number of loop executions needed to have one successful execution is less than 2. Since the number of successful executions is d, the proof follows. $\square$

To tide up the analysis of the expected running time of the algorithm, we use the following:

- If the number of constraints is small ($m \leq C_d$), using the simplex algorithm takes $O(d)^{d/2+O(1)}$ time.
- Otherwise, $m > C_d$. We call the algorithm recursively $O(d)$ times with fewer constraints, the constraints in $R \cup V^*$. The size of R is $r = 2d\sqrt{m}\log m$. The size of V^* after the i^{th} successful iteration is not more then $i\sqrt{m}$. Thus, the number of constraints in the recursive calls is not more than $3d\sqrt{m}\log m$.
- Checking whether a given constraint H is violated by a d-dimensional point v, takes $O(d)$ time. Thus, for a given vertex v, counting how many constraints are violated takes $O(md)$ time. ($O(d)$ time for each of the m constraints.)

The expected running time $T(d,m)$ for this algorithm satisfies

$$T(d,m) \leq \begin{cases} O(d)^{\frac{d}{2}+O(1)} & \text{if } m \leq C_d \\ d[T(d, 3d\sqrt{m}\log m) + O(dm)] & \text{otherwise} \end{cases}$$

After at most $\log\log_d m$ recursive calls, we get to the recursion basis. Therefore, we get

$$T(d,m) = d^{\log\log_d m} O(d)^{\frac{d}{2}+O(1)} + O(d^2 m) =$$

$$= (\log_d m)^{\log d} \cdot O(d)^{\frac{d}{2}+O(1)} + O(d^2 m)$$

This completes the analysis of this algorithm.

Chapter 13

Counting

In this lecture we discuss randomized algorithms for enumeration problems.

13.1 The Class #P

Given a problem instance, we consider the following: *deciding* whether there is a solution, *searching* for a solution, and *enumerating* the number of solutions. In this chapter we concentrate on enumerating the number of solutions. For example, given a satisfiability problem, we may want to enumerate the number of truth assignments that satisfy the given set of clauses.

Clearly, any enumeration problem associated with an NP-complete problem is difficult, and it may even be more difficult than the corresponding decision problem. For example, even if satisfiability could be solved in polynomial time, it does not mean that counting the number of satisfying truth assignments can be done in polynomial time.

Valiant [Va1] defined a new class of polynomial time equivalent problems called #P - complete problems. It is defined in terms of polynomial time relations: $R \subseteq \{0,1\}^* \times \{0,1\}^*$, such that membership in R can be decided in polynomial time, i.e., $R \in P$. The class #P is a class of functions mapping strings to natural numbers :

$$f : \{0,1\}^* \to \mathcal{N}$$

Definition 13.1.1 *Function $f : \{0,1\}^* \to \mathcal{N}$ is in the class #P if for some relation $R \in P$,*

$$f(x) = |\{y : (x,y) \in R\}|$$

A counting problem Y is reducible to problem X if there is a polynomial- time Turing Machine for solving Y when given a subroutine for solving X. X is #P-complete if every problem Y in #P is reducible to X, and X in #P .

The enumeration problem associated with many NP-complete problems is #P-complete . For example, counting the number of truth assignments that satisfy a 3-SAT formula. It turns out that some enumeration problems are #P-complete even when the associated search problems can be solved in polynomial time. For example, counting the number of assignments that satisfy a 2-CNF formula; counting the number of perfect matching in a bipartite graph; counting the number of assignments that satisfy a DNF formula.

We should note that there are some nontrivial enumeration problems that can be solved in polynomial time. For example, counting the number of distinct spanning trees in a graph.

In Section 13.2 we present a randomized polynomial algorithm for approximating the number of satisfying assignments to a DNF formula. In Section 13.3 we show that it is difficult to approximate the number of satisfying assignments of a monotone 2-CNF formula.

13.2 Approximating a DNF formula is easy

A DNF formula is a disjunction of m clauses $C_1 \dots C_m$, where each clause is the conjunction of a subset of literals defined with respect to n boolean variables $x_1 \dots x_n$.

Given a DNF formula $\mathcal{F}$, we want to count the number of assignments that satisfy $\mathcal{F}$. This quantity is denoted by #F . The DNF counting problem is known to be a #P-complete problem [Va1]. Therefore, the best we can hope for in polynomial time is to approximate the number of satisfying assignments.

We first present a sampling algorithm in an abstract setting that will turn out to be useful.

13.2.1 A sampling algorithm

Suppose we have a finite universe U of known size where some of the elements are "good". The set of good elements is denoted by G and we can check whether an element $u \in U$ belongs to G by evaluating a function $f : U \to \{0, 1\}$ that can be efficiently computed. (I.e., $G = \{u \in U | f(u) = 1\}$). Our goal is to to approximate the number of good elements in U and assume we have a method of choosing elements from U uniformly at random.

Definition 13.2.1 *An $(\varepsilon, \delta)-$approximation algorithm is a randomized algorithm which on every input U and f, and for every $\varepsilon > 0, \delta > 0$, outputs a number $\tilde{Y}$ (the approximation to the number of "good" elements), such that*

$$P_r[(1-\varepsilon)\cdot|G| < \tilde{Y} < (1+\varepsilon)\cdot|G|] \geq 1-\delta$$

We use a Monte-Carlo algorithm to obtain the $(\varepsilon, \delta)-$approximation. The i^{th} phase of the Monte-Carlo algorithm consists of randomly choosing $u \in U$, computing $f(u)$, and setting $Y_i \leftarrow f(u)$. There will be N phases. Let $Y = \sum_{i=1}^{N} Y_i$; then the output of the algorithm is $\frac{Y}{N}\cdot|U|$.

Theorem 13.2.1 [Sampling Theorem] *Let μ be the ratio of the "good" elements : $\mu = \frac{|G|}{|U|}$. If $N \geq \frac{1}{\mu}\cdot\frac{2}{\varepsilon^2}\cdot\ln(\frac{2}{\delta})$, then the algorithm described above is an $(\varepsilon, \delta)-$approximation algorithm .*

Proof: The random variables $Y_i, 1 \leq i \leq N$ are

$$Y_i = \begin{cases} 1 & \text{with probability } \mu \\ 0 & \text{with probability } 1-\mu \end{cases}$$

Therefore the random variable Y is (N, μ)-binomially distributed. Such a variable satisfies the next two Chernoff inequalities :

$$P_r[Y \leq (1-\varepsilon)\cdot N\mu] \leq e^{-\varepsilon^2\cdot\frac{N\mu}{2}}$$

$$P_r[Y \geq (1+\varepsilon)\cdot N\mu] \leq e^{-\varepsilon^2\cdot\frac{N\mu}{2}}$$

In order to get an $(\varepsilon, \delta)-$approximation algorithm we require that

$$2e^{-\varepsilon^2\cdot\frac{N\mu}{2}} \leq \delta.$$

or,

$$N \geq \frac{1}{\mu}\cdot\frac{2}{\varepsilon^2}\cdot\ln(\frac{2}{\delta}).$$

$\square$

13.2.2 The approximation scheme

Given a DNF formula $\mathcal{F}$ over n variables, we define a set $U_{\mathcal{F}}$ and a function $f_{\mathcal{F}}$ as follows.

- $U_{\mathcal{F}}$ is the set of all 2^n assignments .
- For every $s \in U_{\mathcal{F}}$, $f_{\mathcal{F}}(s)$ is the truth value of $\mathcal{F}$ under the assignment s.

We may try and apply the sampling algorithm in this setting. However, remember that we require that $N \geq \frac{1}{\mu} \cdot \frac{2}{\varepsilon^2} \cdot \ln(\frac{2}{\delta})$. The problem is that the number of samples, N, is proportional to $\frac{1}{\mu}$. It means that if the actual number of satisfying assignments is small, N would have to be very large. For example, if the number of satisfying assignment is exponentially small, N must be $O(2^n)$.

Karp, Luby, and Madras [KLM] suggested a method for counting the number of satisfying assignments such that a polynomial number of samples will suffices.

Given a clause C_i, let D_i be the set of all of its satisfying assignments, and let D be $\bigcup_{i=1}^{m} D_i$. Clearly $|D_i| = 2^{n-l}$, where l is the number of variables in the clause C_i.

Remark 13.2.1 *Assignment s satisfies a clause C_i for some i iff s satisfies the formula $\mathcal{F}$.*

We would like to compute the cardinality of the set D.

Definition 13.2.2 *U is the set of all pairs (s, i), such that s is an assignment that satisfies the i^{th} clause.*

Next we define a function $f : U \rightarrow \{0, 1\}$.

$$f(s,i) = \begin{cases} 1 & \text{If i is the index of the minimal clause that } s \text{ satisfies.} \\ 0 & \text{Otherwise} \end{cases}$$

Lemma 13.2.1

$$\sum_{(s,i) \in U} f(s,i) = |D|$$

Proof: We can define for each satisfying assignment s the set $B_s = \{(s,j) | \ s \text{ satisfies } C_j\} \subseteq U$. There is precisely one index k for which $f(s,k) = 1$, for all other indices $j > k$, $f(s,j) = 0$. From that we can see that each satisfying assignment s is counted only once. Therefore the sum is exactly $|D|$.

□

We view the set U as our new "universe" and we want to count the number of elements in U for which the function f is non-zero. Let G be the set of these elements from U,

$$G = \{(s,i) | f(s,i) = 1\}.$$

Lemma 13.2.2 $\mu = \frac{|G|}{|U|} \geq \frac{1}{m}$.

Proof: A satisfying assignment s, satisfies at most m clauses, therefore there are at most m elements (s,j) in U, corresponding to each clause C_j.

□

It is clear now that a polynomial number of random samples from U suffice. It remains to establish that we can sample at random from U and that we can compute $f(s,i)$ efficiently.

<u>The Algorithm:</u> (choose an element from U uniformly at random)

1. Choose i at random, $i \in \{1 \cdots m\}$, such that

   $$\Pr[\text{ choose } i] = \frac{|D_i|}{|U|}.$$

   This can be done by choosing $x \in \{1 \cdots |U|\}$. Our i will satisfy

   $$\sum_{j=1}^{i-1} |D_j| < x < \sum_{j=1}^{i} |D_i|.$$

2. Choose uniformly at random $s \in D_i$. This is also easy, the variables belonging to the clause C_i have a fixed value and the other variables can be chosen uniformly over $\{0,1\}$.

Lemma 13.2.3 *An element (s,i) generated by the algorithm is uniformly chosen from U.*

Proof: The probability of choosing i is $\frac{|D_i|}{|U|}$. The probability of choosing assignment s from D_i is $\frac{1}{|D_i|}$. Therefore the probability of each pair $(s,i) \in U$ is $\frac{|D_i|}{|U|} \cdot \frac{1}{|D_i|} = \frac{1}{|U|}$.

□

The approximation algorithm proceeds as follows. For each pair (s,i) generated by the algorithm, compute $f(s,i)$. This is done as follows: For each k, $1 \leq k \leq i$, check whether s satisfies C_k. If there is a k such that s satisfies C_k, we know that $f(s,i) = 0$, otherwise $f(s,i) = 1$. Let Y_j be the random variable denoting the result of the j^{th} sample. The output of the algorithm is $\frac{\sum_{j=1}^{N} Y_j}{N} \cdot |U|$.

By the Sampling Theorem, to obtain an $(\varepsilon,\delta)-$approximation scheme, N should be at least $\frac{1}{\mu} \cdot \frac{2}{\varepsilon^2} \cdot \ln(\frac{2}{\delta})$. Since $\frac{1}{\mu} \leq m$, we can let $N = 2m\ln(\frac{2}{\delta})\frac{1}{\varepsilon^2}$.

The comlexity of the approximation scheme is :

1. The preprocessing to calculate $|D_i|$ and $|U|$ is $O(m \cdot n)$.

2. The complexity of generating a sample is $[O(n) + O(m \cdot n)]$.

Therefore, the total complexity of the algorithm is $O(m \cdot n \cdot N) = O(m^2 \cdot n \cdot \ln(\frac{2}{\delta}) \cdot \frac{1}{\varepsilon^2})$.

Open Question : Is there an $(\varepsilon, 0)$ approximation scheme for the number of assignments satisfying a DNF formula.

13.3 Approximating a monotone 2-CNF formula is hard

Recently, [AS2] and [ALMSS] showed that approximating the MAX-CLIQUE problem below a factor of n^ε is NP-complete for some $\varepsilon > 0$. We will show a reduction of the problem of approximating the cardinality of the maximal clique size of a graph G to the problem of counting satisfying assignments of a monotone 2-CNF formula. A monotone formula is a formula that does not contain negated variables. This is work of D. Zuckerman [Zu].

Given a graph $G = (V,E)$, we define the formula $\mathcal{F}$:

$$\mathcal{F} = \prod_{(i,j) \notin E} (\bar{x}_i \vee \bar{x}_j).$$

Each assignment s can be interpreted as a characteristic vector defining a subset of vertices $V_s = \{v_i \mid x_i = 1 \text{ in } s\}$.

Lemma 13.3.1 *Assignement s satisfies $\mathcal{F}$ iff V_s is a clique in G.*

Proof:

- If V_s is a clique, then for each $(i,j) \notin E$, i and j cannot both belong to the clique. Therefore, either $x_i = 0$ or $x_j = 0$.
- If s satisfies $\mathcal{F}$, then there is an edge connecting any two vertices in V_s. Otherwise, there are two vertices v_1, v_2, such that $x_1 = 1, x_2 = 1$, and $(v_1, v_2) \notin E$. But, then the clause $(\bar{x}_1 \vee \bar{x}_2)$ is not satisfied.

$\square$

This lemma establishes a 1-1 correspondence between cliques and satisfying assignments. Let N be the number of satisfying assignments, and let w be the size of the maximal clique in the graph. We can see that $N \geq 2^w$, because there are at least 2^w cliques corresponding to all subsets of the maximal clique. Hence,

$$\text{number of cliques} \leq \binom{n}{w} + \binom{n}{w-1} + \ldots + \binom{n}{1} \leq n^w.$$

Therefore, we know that $2^w \leq N \leq n^w$, or $\frac{\log N}{\log n} \leq w \leq \log N$.

Theorem 13.3.1 *If the* log *of the number of satisfying assignments to a monotone 2-CNF can be approximated to within a factor of t, then P = NP.*

Proof: If there is a t-approximation to the log of the number of satisfying assignments then there is a $(t \cdot \log n)$-approximation to the maximal clique size in a graph. This implies that P = NP.

$\square$

Part III

Derandomization

Chapter 14

General

A randomized algorithm $\mathcal{A}$ has a probability space (Ω, P) associated with it, where Ω is the sample space and P is some probability measure. Each point in Ω corresponds to a sequence of coin flips performed by the algorithm. We call a point $w \in \Omega$ a *good* point for some input instance I, if $\mathcal{A}(I, w)$ computes the correct solution. We assume that for each I, if w is chosen uniformly at random, then the probability that w is good for I is sufficiently high.

A natural approach towards derandomizing algorithms is to find a method for searching the associated sample space Ω for a good point w with respect to a given input instance I. Given such a point w, the algorithm $\mathcal{A}(I, w)$ becomes a deterministic algorithm and it is guaranteed to find the correct solution. The problem faced in searching the sample space is that it is generally exponential in size.

Two general methods for searching the sample space have emerged in recent years and will be described in forthcoming lectures.

The first approach is that of *the method of conditional probabilities* [Sp, Ra]. This method was introduced by Spencer [Sp] with the aim of converting probabilistic proofs of existence of combinatorial structures into efficient deterministic algorithms for actually constructing these structures. The idea is to perform a *binary search* of the sample space Ω for a good point. (In Spencer's case, points in Ω correspond to different combinatorial structures and a good point corresponds to a structure which satisfies some desired properties). At each step of the binary search, the current sample space is split into two equal halves and the *conditional probability* of obtaining a good point is computed for each half. The search is then restricted to the half where the conditional probability is higher. The search terminates when only one sample point, which must be a good point, remains. This method is applicable to large sample spaces since it requires $\log |\Omega|$ steps only.

A different approach is to conduct an exhaustive search of the sample space. Since there is a good point in Ω for each possible input, trying every $w \in \Omega$ is guaranteed to yield a good point. The problem with this approach is that in general, the size of the sample space is exponential in the size of the input. This difficulty was overcome in certain cases by constructing a (different) polynomial-sized sample space. One way of constructing the new sample space is by showing that the probabilistic choices of the randomized algorithm are only required to be k-wise independent. (Then, a sample space of size $O(n^k)$ suffices). An example in which this approach is used is computing in NC a maximal independent set in a graph. Later in the course we will discuss other methods for constructing small sample spaces, e.g., small bias probability spaces. An advantage of the reduction in size of the sample space is that it is especially suited for parallel algorithms because each point in Ω can be checked independently of the other points.

We first present a result of Adleman [Ad] showing that the sample space Ω associated with a randomized algorithm always contains a *polynomial-sized* subspace which has a good point for each possible input instance. (In fact, it shows that RP $\subseteq$ P/poly). However, this result is highly non-constructive and it cannot be used to actually derandomize algorithms.

Theorem 14.0.2 *There exists a set $S \subset \{0,1\}^k$ of size n such that for every input of size n there is at least one "good" sequence in S.*

Proof: Let $A_{i,j}$ be a $2^n \times 2^k$ matrix where the rows correspond to the inputs and the columns correspond to the points in the sample space.

$$A_{i,j} = \begin{cases} 1 & \text{if point } j \in \Omega \text{ is "good" for input } i \\ 0 & \text{otherwise} \end{cases}$$

The set S is constructed iteratively. Initially, S is empty. Clearly, each row in A has at least 2^{k-1} "1"s. Thus, for at least one column j in A, more than half of its entries are "1", i.e., there exists a point $j \in \Omega$ that is "good" for at least half of the inputs.

We add j to S and construct A' by discarding all rows in A that correspond to inputs for which j is "good". The above argument holds for A' as well and thus, matrix A is exhausted in at most n iterations.

Another way of stating this argument is the following: Let F_I denote the event that n independent runs of algorithm A on input I failed. Obviously, $\Pr(F_I) < \left(\frac{1}{2}\right)^n$. Therefore,

$$\Pr\left(\bigcup_I F_I\right) \leq \sum_I \Pr(F_I) < 1 \tag{14.1}$$

In other words, we have constructed a new algorithm A', where running it on input I is the same as n independent runs of algorithm A on input I. By (9.1), The sample space of A' contains a point that is "good" for all inputs I. $\square$

Chapter 15

The Method of Conditional Probabilities

In this chapter we show several examples in which the method of conditional probabilities can be applied to derandomize algorithms. As previously mentioned, the idea is to perform a binary search of the sample space associated with the random variables so as to find a good point. In a typical setting, the random variables are $\{0, 1\}$ and the binary search is implemented as follows. The random variables are considered one-by-one. At each step of the algorithm, the value of some of the variables has already been set, one variable is currently being considered, and the value of the rest is chosen at random. At each step we will compute the (conditional) probability of finding a good point in the sample space if the variable considered is set to 0 and if it is set to 1. The variable in consideration is then set to the value where the conditional probability is higher and the search terminates when setting the value of the last variable. The main difficulty in implementing the method is in computing the conditional probabilities.

15.1 Computing a large cut

Given a graph $G(V, E)$, a *cut* C is a partition of the vertices into two sets. The weight of the cut ($|C|$) is defined to be the number of edges adjacent to vertices in both sets. In Section 2.4 we showed that in any graph there exists a cut C such that $|C| \geq \frac{|E|}{2}$. We now show how to compute the cut C.

Notice that the probabilistic proof of existence in Section 2.4 can be easily converted into a randomized algorithm. Let $X_i \in \{0,1\}$ denote the label of vertex i. The algorithm assigns labels to the vertices at random with probability: $\Pr[X_i = 1] = \Pr[X_i = 0] = \frac{1}{2}$. It follows from the analysis in Section 2.4 that the expected weight of a cut generated by the algorithm is $|E|/2$.

15.1.1 Derandomization

We consider the vertices one-by-one and assign them labels $\epsilon_i = \{0,1\}$ $1 \le i \le n$. Clearly, label ϵ_1 can be chosen arbitrarily. Suppose the labels $\epsilon_1, \epsilon_2, \ldots, \epsilon_{i-1}$ have already been determined, it is left to show how ϵ_i is determined.

After phase $i-1$, (determining labels for the first $i-1$ vertices), the edges can be partitioned into 3 sets, according to the labels of the vertices adjacent to each edge.

1. Edges that are in the cut (their number is denoted by $C^{d,i-1}$).
2. Edges that are not in the cut.
3. Edges that are undetermined yet (their number is denoted by $C^{r,i-1}$).

Let $C^{i-1} = C^{r,i-1} + C^{d,i-1}$ be a random variable. Note that

$$E[C^{i-1}] = E[C^{r,i-1}] + C^{d,i-1}.$$

Determining the value of ϵ_i can only affect edges that are adjacent to vertices that have already been labeled. (An edge adjacent to an unlabeled vertex has probability $\frac{1}{2}$ of joining the cut). Therefore:

$$E[C^{r,i-1}] = \frac{1}{2} \times \left(E[C^{r,i-1} \mid X_i = 0] + E[C^{r,i-1} \mid X_i = 1] \right)$$

and hence

$$E[C^{r,i-1}] \le \max\left(E[C^{r,i-1} \mid X_i = 0], E[C^{r,i-1} \mid X_i = 1] \right). \tag{15.1}$$

Therefore, ϵ_i can be chosen such that the right hand side of Inequality (15.1) is maximized, yielding,

$$\frac{|E|}{2} = E[C^0] \le E[C^1] \le \ldots \le E[C^n] = |C|.$$

It still remains to show how to compute the conditional probabilities at each step. For each edge adjacent to v_i, if it is adjacent to vertex v_j, where $j > i$, its contribution to $E[C^{r,i-1} \mid X_i = \epsilon_i]$ is $\frac{1}{2}$, otherwise it is either 0 or 1. Hence, if the labeling of the majority of v_i's neighbors is ϵ, then setting $X_i = 1 - \epsilon$ would maximize the right hand side of inequality 15.1.

15.2 The set balancing problem

In this section we demonstrate the usage of the method of conditional probabilities for the *set balancing problem* which is defined formally as follows:

Input: A collection of subsets $\mathcal{S} = \{S_1, S_2, \cdots S_m\}$ of a base set $B = \{b_1, b_2, \cdots b_n\}$ such that each $S_i \subseteq B$.

Output: A partition of B into B^0 and B^1 such that for some small function $f(n, m)$ and for all S_i,

$$\Big||B^0 \cap S_i| - |B^1 \cap S_i|\Big| \leq f(n, m)$$

The aim is to minimize the value of $f(n, m)$ or, equivalently, find a partition of B such that each subset $S_i \in \mathcal{S}$ has roughly the same number of elements in B^0 and B^1. Initially, we will only concern ourselves with the case where $m = n$ and each set $S_i \in \mathcal{S}$ has cardinality δ. The bounds extend immediately to the case where the cardinality of $\mathcal{S}$ and each $S_i \in \mathcal{S}$ is arbitrary.

Definition 15.2.1 *A 2-coloring of B is an n-bit vector $\vec{C} = (C_1, C_2, \cdots C_n)$ such that $C_i \in \{-1, 1\}$ is the color of $b_i \in B$. It follows that*

$$B^{-1} = \{b_i \in B : C_i = -1\}$$

$$B^1 = \{b_j \in B : C_j = 1\}$$

Definition 15.2.2 *The partition of a set $S_i \in \mathcal{S}$ induced by a 2-coloring is given by*

$$S_i^{-1} = S_i \cap B^{-1}$$

$$S_i^1 = S_i \cap B^1$$

The following notions of *discrepancy* will help us measure the quality of a solution to the set balancing problem. The discrepancy for a set S_i is essentially the difference in the sizes of the two sets S_i^{-1} and S_i^1.

Definition 15.2.3 *The* set discrepancy *of a set $S_i \in \mathcal{S}$ with respect to a 2-coloring $\vec{C}$ is given by*

$$\Delta(S_i, \vec{C}) = \sum_{b_j \in S_i} C_j$$

Definition 15.2.4 *The* family discrepancy *of $\mathcal{S}$ with respect to a 2-coloring $\vec{C}$ is given by*

$$\Delta(\mathcal{S}, \vec{C}) = \max_{S_i \in \mathcal{S}} \Delta(S_i, \vec{C})$$

Spencer [Sp1] has shown that for each family $\mathcal{S}$ there exists a 2-coloring $\vec{C}$ such that $\Delta(\mathcal{S}, \vec{C}) \leq 6\sqrt{n}$. This result is the best possible up to constant factors but it has the drawback of being non-constructive, i.e. does not even imply a randomized algorithm.

In the following we present a randomized algorithm which guarantees a 2-coloring $\vec{C}$ with discrepancy $\Delta = \sqrt{2\delta \ln 2n}$. Then, the method of conditional probabilities is applied to derandomize the algorithm. In this case computing the conditional probabilities is more complicated than in the previous example.

15.2.1 The randomized algorithm

The algorithm simply assigns each element b_i a color 1 or -1 with probability $\frac{1}{2}$. In the following theorem it is shown that the probability of obtaining a good coloring is at least $\frac{1}{2}$.

Theorem 15.2.1 *There exists a 2-coloring with $\Delta = \sqrt{2\delta \ln 2n}$.*

Proof: Let X_i be a random variable denoting the color of element b_i where $\Pr[X_i = -1] = \Pr[X_i = 1] = \frac{1}{2}$ and let T_i be the random variable denoting the discrepancy in subset S_i. Note that $E[T_i] = 0$ and that each T_i is binomially distributed $B(\delta, \frac{1}{2})$. Using Chernoff's inequality we can derive the following bound

$$\Pr[|\sum_{b_j \in S_i} X_j| > a] \leq e^{-\frac{a^2}{2\delta}}.$$

Since there are n subsets it is sufficient that $e^{-\frac{a^2}{2\delta}} \leq \frac{1}{2n}$. If $a > \sqrt{2\delta \ln 2n}$, then the probability of a bad coloring is less than $\frac{1}{2}$. □

15.2.2 Derandomization

The derandomization process can be pictured by a binary tree. Assume we assign colors to the elements in the order $b_1, b_2, \ldots, b_n$. The nodes of the tree correspond to elements in B and the edges correspond to assignments of colors to the elements. The root of the binary tree corresponds to element b_1 and the two edges adjacent to the root correspond to the two possible assignments of a color to b_1. The two children of b_1 correspond to element b_2, where in one subtree $b_1 = 1$ and in the other, $b_1 = -1$. There is a 1-1 correspondence between the leaves of the binary tree and the 2^n possible colorings of B. In other words, each coloring of B is a path from the root to a leaf. We call a leaf good if the discrepancy of the coloring is not larger than $\sqrt{2\delta \ln 2n}$.

Let P_v denote the probability of obtaining a bad coloring at node v in the tree, i.e., P_v is the percentage of not good leaves in the subtree of node v. Note that

$$P_v = \frac{P_{v_1} + P_{v_{-1}}}{2} \Rightarrow P_v \geq \min(P_{v_1}, P_{v_{-1}}).$$

where P_{v_1} ($P_{v_{-1}}$) is the probability of a bad coloring at node v assuming that v's color is 1 (-1). If, at each node v, the probability P_v could be computed, then obviously a good coloring could also be computed by choosing at each node of the tree a subtree that contains at least one good leaf. (Notice that the probability P_v is conditioned on the path chosen from the root to v). However, it is not known how to compute at each node v the precise probability of obtaining a good coloring. The main reason for that is that in the proof of Theorem 15.2.1, we bound from above the probability of a union of events by the sum of the probabilities of the events. It may happen that even if this estimate guarantees at the root of the tree that a good coloring exists, in some internal node v of the tree, the probability of obtaining a good coloring will vanish. (Although there is a good leaf in the subtree of v!)

To overcome this difficulty, we introduce the notion of a *pessimistic estimator* of the probability of obtaining a bad coloring in the subtree of v. A pessimistic estimator, $\widehat{P}_v$, must have the following properties:

1. $\widehat{P}_{\text{root}} < 1$.
2. $\forall v,\ P_v \leq \widehat{P}_v$.
3. $\forall v,\ \widehat{P}_v \geq \min(\widehat{P}_{v_1}, \widehat{P}_{v_{-1}})$.
4. The estimators can be computed in polynomial time.

Given the estimators, the derandomization is done as follows: both $\widehat{P}_{v_1}, \widehat{P}_{v_{-1}}$ are computed and the color assignment which minimizes the estimated probability is chosen. This process always succeeds since

$$\min(\widehat{P}_{v_1}, \widehat{P}_{v_{-1}}) \leq \widehat{P}_v \leq \widehat{P}_{\text{root}} < 1.$$

It is left to show how these estimators are constructed and that they enjoy the desired properties. Chernoff's inequality states that:

$$\Pr[\mid T_i \mid \geq \Delta] \leq \frac{E[e^{tT_i}]}{e^{t\Delta}}$$

First, an estimator to the root is constructed.

$$\widehat{P}_{\text{root}} = \sum_{i=1}^{n} \frac{E[e^{tT_i}]}{e^{t\Delta}}$$

It follows from Theorem 15.2.1 that there exists a t for which the above sum is smaller than 1.

Now we will show how to construct an estimator for an internal node. Suppose $X_1 = \epsilon_1, X_2 = \epsilon_2, \ldots, X_j = \epsilon_j$ are already determined. Assume that node v determines X_{j+1}. For every subset S_i, denote by T_i^j the random variable which is the sum of the colors of the elements in S_i after X_j was set. Note that

$$T_i^j = T_{i,r}^j + T_{i,d}^j$$

where $T_{i,r}^j$ is the contribution of the elements for which colors have not been determined yet, and $T_{i,d}^j$ is the contribution of the already colored elements in S_i.

$$\begin{aligned}
\Pr[T_i^j \geq \Delta] &= \Pr[T_{i,r}^j + T_{i,d}^j \geq \Delta] \\
&= \Pr[T_{i,r}^j \geq \Delta - T_{i,d}^j] \\
&\leq \frac{E[e^{tT_{i,r}^j}]}{e^{t(\Delta - T_{i,d}^j)}}
\end{aligned}$$

The estimator at node v will be the sum of the bounds for the different subsets.

$$\widehat{P}_v = \sum_{i=1}^{n} \frac{E[e^{tT_{i,r}^j}]}{e^{t(\Delta - T_{i,d}^j)}}$$

Note that:

$$\begin{aligned}
\widehat{P}_v &= \sum_{i=1}^{n} \frac{1}{2} \times \left(\frac{E[e^{t(T_{i,r}^j+1)}]}{e^{t(\Delta - T_{i,d}^j)}} + \frac{E[e^{t(T_{i,r}^j-1)}]}{e^{t(\Delta - T_{i,d}^j)}} \right) \\
&= \frac{1}{2} \times \left(\sum_{i=1}^{n} \frac{E[e^{tT_{i,r}^j}]}{e^{t(\Delta - T_{i,d}^j)-1}} + \sum_{i=1}^{n} \frac{E[e^{tT_{i,r}^j}]}{e^{t(\Delta - T_{i,d}^j)+1}} \right) \\
&= \frac{\widehat{P}_{v_1}}{2} + \frac{\widehat{P}_{v_{-1}}}{2}
\end{aligned}$$

and thus,

$$\widehat{P}_v \geq \min(\widehat{P}_{v_1}, \widehat{P}_{v_{-1}}).$$

Note also that it is easy to compute the estimators. Thus, we have shown that they have the required properties.

15.3 Lattice approximation

Informally, we are given a point p in an n-dimensional space and wish to find a *lattice* point q, i.e., with integer coordinates, such that q is a good "approximation" to p. We say that q is a good approximation to p if the vector $p - q$ has a small inner product with each vector in a given collection of vectors, say the rows of some matrix A. It is not very hard to see that this problem is related to that of solving a linear programming relaxation of an integer program (see Lovász [Lo]), and then "rounding" the solution to obtain a near-optimum solution to the original integer program. More formally the problem may be defined as follows:

Input: An $n \times m$ matrix A, with each entry $A_{i,j} \in [0,1]$; a vector $\vec{p} \in [0,1]^m$ and $\vec{c} \in \mathcal{R}^n$ such that, $A\vec{p} = \vec{c}$.

Output: An m-bit vector $\vec{q} \in \{0,1\}^m$ such that it minimizes the value of the error in the approximation which is given by $|| \, |A \cdot \vec{p} \, - \, A \cdot \vec{q}| \, ||_\infty$. We denote the error vector by Δ.

Note that, for a vector $\vec{v}$, the expression $||\vec{v}||_\infty$ denotes the L_∞ norm. The set balancing problem is a special case of the lattice approximation problem where each $p_i = \frac{1}{2}$ and the matrix A is the incidence matrix of subsets and elements.

15.3.1 Randomized rounding

Raghavan [Ra] used the *randomized rounding* technique to obtain a vector $\vec{q}$ such that Δ_i, the i^{th} component of the error vector Δ, is bounded by $\sqrt{2c_i \log n}$.

The vector $\vec{p}$ is rounded to $\vec{q}$ as follows:

$$q_i = \begin{cases} 1 & \text{with probability } p_i \\ 0 & \text{with probability } 1 - p_i \end{cases}$$

The following claim bounds $\Delta_i = |\sum_{j=1}^m A_{i,j}(p_j - q_j)|$.

Claim 15.3.1 *The probability that $\Delta_i > \sqrt{2c_i \log n}$ is strictly smaller than $\frac{1}{n}$.*

Proof: Let $S_i = (Aq)_i$ be a random variable. Notice that S_i is the sum of independent random variables and that $E[S_i] = c_i$ since $E[q_i] = p_i$. We will show that

$$\Pr\left[\Delta_i > \sqrt{c_i \log n}\right] < \frac{1}{n}.$$

We use a variant of Chernoff's inequality:

$$\Pr\left[|S_i - E[S_i]| > \delta E[S_i]\right] < e^{-\frac{\delta^2 E[S_i]}{2}} \tag{15.2}$$

Since we want that $|S_i - c_i| \le \sqrt{2c_i \log n}$, it should hold that $\sqrt{2c_i \log n} < \delta \cdot c_i$. Using Inequality (15.2) and substituting $\delta > \sqrt{\frac{2\log n}{c_i}}$, we get

$$\Pr\left[\Delta_i > \sqrt{c_i \log n}\right] < e^{-\left(\frac{2\log n}{c_i}\right)\frac{c_i}{2}} = \frac{1}{n}.$$

$\square$

Hence, there is a positive probability that for all i, $\Delta_i \le \sqrt{2c_i \log n}$.

The randomized rounding method can be derandomized using the method of conditional probabilities similarly to the derandomization of the set balancing algorithm outlined in the previous section. Randomized rounding has found applications in approximation algorithms for problems such as multi-commodity flow.

15.4 Limitation on parallelization

An interesting question is whether the general form described in the previous section is amenable to parallelization. In other words, what is the speedup achievable when many processors are available.

(We note that the set balancing problem (and lattice approximation) can be solved efficiently in parallel by combining the method of conditional probabilities with limited independence. See [Lu2, BR, MNN] for more details).

To answer this question, we consider the following concrete complexity type scenario. We are given a discrete space Ω in which the points are each labeled either *good* or *bad*. For any subset $\Omega' \subset \Omega$, the Boolean function $P(\Omega')$ tells whether there are good points in Ω' or not. The problem is to identify any one good point in Ω under the following computation model. We have q processors, each processor can compute $P(\Omega')$ in one time-unit for any $\Omega' \subset \Omega$. After each round of computation, the processors can share all the information they have gained so far and decide on the queries of the next round. However, we assume that the only access the processors have to Ω is through a "black-box" computation of P, and we completely ignore the complexity of deciding which queries to ask. This is an instance of a family of problems called *group testing*.

It is easy to see that the sequential complexity (one processor) of finding a good point under such conditions is $\log|\Omega|$. The next theorem tells us that having q processors hardly helps.

Theorem 15.4.1 *For all q, the number of rounds required to find a good point in Ω using q processors is at least $\frac{\log|\Omega|}{\log(q+1)}$.*

The proof of the theorem follows immediately from an adversary argument similar to the one given in Theorem 1 of [KUW2].

Notice, however, that when the conditional probability of finding a good point in a subset Ω' is computed, we are in fact computing the number of good points in Ω'. It is natural to ask the following: If the function $P(\Omega')$ is strengthened, i.e., it tells us the exact number of good points in Ω', can we prove a bound similar to the one in Theorem 15.4.1? This is still an open question.

Chapter 16

Limited Independence

In this chapter and in the following we discuss a different approach to derandomization. The idea is to conduct an exhaustive search of the sample space associated with a randomized algorithm so as to find a "good" point. To make this approach a feasible one, it is necessary first to reduce the size of the sample space to a polynomial. An important technique for achieving this goal is showing that the probabilistic choices made by the randomized algorithm are only required to be k-wise independent for some constant k. We provide efficient constructions for generating k-wise independent random variables; the size of the sample space is roughly $O(n^k)$.

This idea can be demonstrated by the following simple example. In previous lectures we discussed the problem of computing a large cut (see Sections 2.4 and 15.1). Note that the assumption that the random choices are completely independent is too strong. If the random labels of the vertices are restricted to be only pairwise independent, the sample space is still guaranteed to contain a large cut!

We first present two constructions of k-wise independent random variables. The first construction generates random variables that are uniform on $\{0, 1\}$. The second construction can generate any $\{0, 1\}$ random variables.

We demonstrate the idea of using limited independence in the problem of computing in parallel a maximal independent set in a graph.

16.1 Constructing k-wise independent uniform random variables

This construction was suggested by [ABI]. Assume we are given m vectors, $L_1, L_2, \ldots, L_m \in \{0,1\}^l$ such that any k vectors are linearly independent over $GF(2)$. Given such vectors, the construction of the k-wise independent random variables is as follows: a vector $R \in \{0,1\}^l$ is chosen uniformly at random. The random variable X_i is simply set to be $\langle L_i, R\rangle$ where $\langle L_i, R\rangle = \sum_{j=1}^{l} L_{i,j} r_j (\mod 2)$, is the inner product in $GF(2)$.

Claim 16.1.1 *The random variables $X = X_1, X_2, \ldots, X_m$ are k-wise independent.*

Proof: Given any arbitrary subset $X_{i_1}, X_{i_2}, \ldots, X_{i_k}$ we show that

$$\Pr[X_{i_1} = d_1, X_{i_2} = d_2, \ldots, X_{i_k} = d_k] = \Pr[X_{i_1} = d_1]\cdot\Pr[X_{i_2} = d_2]\cdot\cdots\cdot\Pr[X_{i_k} = d_k] = \left(\frac{1}{2}\right)^k$$

The above algorithm for generating k-wise independent random variables can be stated as follows. Let H be an $m \times l$ matrix where each row i, $1 \le i \le m$ is L_i. Then, $X = H \cdot R$.

Suppose we restrict the matrix H to rows $i_1, \ldots, i_k$ and substitute the value d_j for each variable X_{i_j}, $1 \le j \le k$. We now regard vector R as our "unknown". The probability of the event $X_{i_1} = d_1, X_{i_2} = d_2, \ldots, X_{i_k} = d_k$ is precisely the percentage of vectors R that satisfy $X = H \cdot R$. Since the rows in the restricted matrix are linearly independent, there are $l - k$ "free" coordinates in the vector R and hence 2^{l-k} possible values for R. Therefore,

$$\Pr[X_{i_1} = d_1, X_{i_2} = d_2, \ldots, X_{i_k} = d_k] = \frac{2^{l-k}}{2^l} = \left(\frac{1}{2}\right)^k$$

□

We now show how to generate vectors that are k-wise linearly independent by using constructions from coding theory.

Definition 16.1.1 *A code $C : \{0,1\}^k \to \{0,1\}^n$ is a mapping of an* information word *$u \in \{0,1\}^k$ to a* codeword *$x \in \{0,1\}^n$ such that $k \le n$.*

Definition 16.1.2 *The (Hamming) distance between two $\{0,1\}^n$ vectors $\vec{x} = x_1, x_2, \ldots, x_n$ and $\vec{y} = y_1, y_2, \ldots, y_n$ is the number of places where they differ, and is denoted by $dist(\vec{x}, \vec{y})$.*

Definition 16.1.3 *The (Hamming) weight of a vector $\vec{x} = x_1, x_2, \ldots, x_n$ is the number of nonzero x_i's, and is denoted by $wt(\vec{x})$.*

Definition 16.1.4 *The* minimum distance d *of a code* C *is the minimum Hamming distance taken over all pairs of codewords.*

Lemma 16.1.1 *A code with minimum distance* d *can correct* $\lfloor \frac{d}{2} \rfloor$ *errors.*

Let $GF(q)$ where q is a prime number denote a finite field and $V(n,q)$ be the n-dimensional vector space $(GF(q))^n$.

Definition 16.1.5 *A linear code over* $GF(q)$ *is a subspace of* $V(n,q)$ *for some* $n > 0$.

Definition 16.1.6 *A linear code of length* n *and subspace dimension* k *is called an* $[n,k]$ *code.*

Definition 16.1.7 *A* $k \times n$ *matrix* G *whose rows form a basis of a* $[n,k]$ *code* C *is called a generator matrix of* C

Given an information word u, the corresponding codeword is defined to be $u \cdot G$. Another important matrix is H, the parity check matrix. The following lemma gives the relation between a linear code and its associated parity check matrix.

Claim 16.1.2 *Given an* $[n,k]$ *code* C*, there exists an* $n-k \times n$ *matrix* H *(the parity check matrix) such that*

$$\left\{ x \in C \Leftrightarrow Hx^T = 0 \pmod{2} \right\}$$

Definition 16.1.8 *An* $[n,k]$ *code with minimum distance* d *is called an* $[n,k,d]$ *code.*

A nice property of $[n,k,d]$ linear codes is that the minimum weight of a codeword is equal to the minimum distance d. This follows immediately from the fact that a linear code is also a subspace and therefore the vector $\vec{0}$ must belong to it.

Claim 16.1.3 *Let* C *be an* $[n,k,d]$ *code with parity check matrix* H*. Then, every* $d-1$ *columns of* H *are linearly independent.*

Proof: Assume that there are columns $i_1, \ldots, i_l$ of H where $l < d-1$, that are linearly dependent. I.e.,

$$\sum_{j=1}^{l} (H)_{i_j} = 0.$$

Define the word $v = v_1, \ldots, v_n$ such that

$$v_j = \begin{cases} 1 & j \in i_1, \ldots, i_l \\ 0 & \text{otherwise} \end{cases}$$

Clearly v is a codeword since $Hv^T = 0$. But, $wt(v) = l < d-1$ which is a contradiction. □

The aforementioned construction of k-wise independent random variables can be achieved using any linear error correcting code. We now describe a construction based on the binary BCH codes. (See [AS1, MS]).

Suppose $n = 2^d - 1$ and $k = 2t+1$. Let $y_1, \ldots, y_n$ be the n nonzero elements of the field GF(2^d) represented as column vectors of length d over GF(2). The $1 + dt \times n$ parity check matrix H of the extended binary BCH code of length n and designated distance $2t+2$ is the following:

$$\begin{pmatrix} 1 & 1 & \ldots & 1 \\ y_1 & y_2 & \ldots & y_n \\ y_1^3 & y_2^3 & \ldots & y_n^3 \\ \vdots & \vdots & & \vdots \\ y_1^{2t-1} & y_2^{2t-1} & \ldots & y_n^{2t-1} \end{pmatrix}$$

We have already proved that any $2t+1$ columns in H are independent. Hence, by this construction, vectors $L_1, \ldots, L_n$ that are k-wise independent belong to $\{0,1\}^l$ where $l = (t+1) \cdot d = (t+1) \cdot \log n$).

16.2 Constructing k-wise independent arbitrary random variables

The construction outlined in the previous section generates uniform $\{0,1\}$ random variables. We now show how to generate d-wise independent random variables that are arbitrarily distributed.

Definition 16.2.1 *Let $X, \widehat{X}$ be two random variables. We define the distance between the distributions of X and $\widehat{X}$ to be:*

$$dist(X, \widehat{X}) = \sup_{x \in R}(\mid \Pr(X = x) - \Pr(\widehat{X} = x) \mid).$$

Theorem 16.2.1 *Let $X_1, X_2, \ldots, X_n$ be random variables such that:*

$$\Pr[X_i = \epsilon_i] = \begin{cases} p_i & \epsilon_i = 1 \\ 1 - p_i & \epsilon_i = 0 \end{cases}$$

Let further $n \leq p \leq 2n$ be a prime and $d \geq 1$. Then there exists a uniform probability space (Ω, P) over a sample space of size $\mid \Omega \mid = p^d$ and random variables $\widehat{X}_1, \widehat{X}_2, \ldots, \widehat{X}_n$ over (Ω, P) such that $dist(X_i, \widehat{X}_i) \leq \frac{1}{p}$ for $1 \leq i \leq n$ and the variables $\widehat{X}_i$ are d-wise independent.

Proof: The existence of p is guaranteed by the Density Theorem of prime numbers. Let $F = GF(p)$ be a finite field. Our sample space Ω will correspond to the set of p^d polynomials of degree $\leq d - 1$ over F. The random variable $\widehat{X}_i$ is defined as follows: given a randomly chosen polynomial $q = q_1 + q_2 x + q_3 x^2 + \ldots + q_d^{d-1} x^d$,

$$\widehat{X_i} = \begin{cases} 1 & \frac{q(i)}{p} > p_i \\ 0 & \text{otherwise} \end{cases}$$

It is easy to see that $\Pr[q(i) = c] = \frac{1}{p}$ for every $c \in GF(p)$. Hence, the rounding can generate an error of at most $\frac{1}{p}$ and therefore $dist(X_i, \widehat{X}_i) \leq \frac{1}{p}$.

It is left to show that the random variables are d-wise independent. We define new random variables $Y_1, \ldots, Y_n$ as follows: random variable Y_i is equal to $q(i)$ where q is a randomly chosen polynomial (as before). We prove the stronger assertion that $Y_1, \ldots, Y_n$ are d-wise independent. Let $Y_{i_1}, Y_{i_2}, \ldots, Y_{i_d}$ be any subset of the random variables. Also, let $q_1, \ldots, q_d$ denote the coefficients of the randomly chosen polynomial q. The random variables are constructed by multiplying an $n \times d$ matrix H by the vector defined by the coefficients of q, i.e.,

$$\begin{pmatrix} 1 & i_1 & \ldots & i_1^{d-1} \\ 1 & i_2 & \ldots & i_2^{d-1} \\ & & \ddots & \\ 1 & i_d & \ldots & i_d^{d-1} \end{pmatrix} \begin{pmatrix} q_1 \\ q_2 \\ \vdots \\ q_d \end{pmatrix} = \begin{pmatrix} Y_{i_1} \\ Y_{i_2} \\ \vdots \\ Y_{i_n} \end{pmatrix}$$

Note that H is a Vandermonde matrix and hence nonsingular. Therefore, for any setting $(d_1, \ldots, d_k)$ of the variables $Y_{i_1}, Y_{i_2}, \ldots, Y_{i_d}$, there is a unique polynomial q that will generate this setting. Hence,

$$\Pr\left[Y_{i_1} = d_1, \ldots, Y_{i_k} = d_k\right] = \left(\frac{1}{q^d}\right)$$

Implying that the random variables are d-wise independent. □

A natural question that arises is whether this construction is optimal. In the following lemma a lower bound on the size of the sample space is given. A variable is *almost constant* if it takes a single value with probability 1.

Let $m(n, d)$ denote the following sum of binomial coefficients:

$$m(n,d) = \begin{cases} \sum_{j=0}^{d/2} \binom{n}{j} & \text{if } d \text{ is even} \\ \sum_{j=0}^{(d-1)/2} \binom{n}{j} + \binom{n-1}{\frac{d-1}{2}} & \text{if } d \text{ is odd} \end{cases}$$

Theorem 16.2.2 *Assume that the random variables $X_1, \ldots, X_n$ over the sample space Ω are d-wise independent and not almost constant. Then, $\mid \Omega \mid \geq m(n, d)$.*

Proof: We can assume w.l.o.g. that $\forall i,\ E[X_i] = 0$. For any subset S of $\{1, 2, \ldots, n\}$ let $\alpha_S = \prod_{i \in S} X_i$. By the d-wise independence of X_i.

$$E[\alpha_S \alpha_T] = \begin{cases} \text{positive} & \text{if } S = T \text{ and } \mid S \mid \leq d \\ 0 & \text{if } S \neq T \text{ and } \mid S \cup T \mid \leq d \end{cases}$$

Let now $S_1, \ldots, S_m$ be subsets of $\{1, 2, \ldots, n\}$ such that $\mid S \cup T \mid \leq d$ for every i, j, where $m = m(n, d)$. (Take all sets of size $\leq d/2$ and, if d is odd, add those $(d+1)/2$ containing element 1.)

It is left to show that the functions α_{S_i} for $i = 1, \ldots, m(n, d)$ are linearly independent. This implies the desired inequality. Suppose some linear combination $\sum_{i=1}^{m} c_i \alpha_{S_i} = 0$. Then for any j, multiplying by α_{S_j}, we get that,

$$0 = E\left[\sum_{i=1}^{m} c_i \alpha_{S_i}\right] = \sum_{i=1}^{m} c_i E\left[\alpha_{S_i} \alpha_{S_j}\right] = c_j E\left[\alpha_j^2\right].$$

Thus, for all j, the coefficients c_j are zero. □

16.3 An application - computing a maximal independent set

We now demonstrate the usage of limited independence in the problem of computing a maximal independent set in a graph in parallel. We first present a randomized algorithm for the problem that uses complete independence. Then we show that if the random choices are only pairwise independent, there is still a nonzero probability of success. We conclude by derandomizing the algorithm.

16.3.1 Problem definition

Definition 16.3.1 *Let $G = (V, E)$ be an undirected graph and let $|V| = n$.* A maximal independent set *in G is a set of vertices $MIS \subseteq V$ that does not contain adjacent vertices and is not contained in any other independent set.*

The problem can be solved sequentially in the following way : begin with an empty set and examine the vertices $v \in V$ in some arbitrary order. Each vertex is added to the MIS iff it is not a neighbor of a vertex already in the MIS. This algorithm seems to be inherently sequential and therefore a different approach is needed for computing an MIS in NC. The first NC algorithm was given in 1983 by Karp and Wigderson [KW]. A simpler algorithm was proposed by Luby in 1985 [Lu1] and a similar algorithm was also given by Alon et al. [ABI]. Our exposition follows Alon et al. Conceptually, there are three steps:

- Prove that a parallel randomized algorithm computes an MIS with high probability.
- Show that the analysis of the randomized algorithm is valid even if the random variables are pairwise independent.
- Derandomize the algorithm : random variables that are only pairwise independent require a small sample space (polynomial size).

16.3.2 The randomized algorithm

We now present a randomized algorithm for computing a maximal independent set.

$MIS = \emptyset$

Repeat

1. Each vertex $v \in V$ marks itself with probability $1/2d(v)$.
2. If two neighbors marked themselves, the vertex with smaller degree unmarks itself. (Ties are broken arbitrarily).
3. $MIS \leftarrow MIS \cup \{v | v \text{ is marked}\}$.
4. $V \leftarrow V \setminus \{v | v \in MIS \vee \exists u \in MIS, (u, v) \in E\}$.

Until $V = \emptyset$

Note that $d(v)$, the degree of v, is calculated in the induced graph.

16.3.3 Analysis

We show that the expected number of iterations is $O(\log n)$. We do that by showing that the expected number of edges deleted in each iteration is a constant fraction of the total number of edges at the beginning of the iteration. Therefore, the following analysis applies to a single iteration.

We first assume that all the random choices made in Step 1 are completely independent. We say that a vertex v is *bad* if the degree of more then $2/3$ of its neighbors is bigger than $d(v)$. A vertex which is not bad is called *good*.

Lemma 16.3.1 *If $u \in V$ is a good vertex, then the probability p that one of its neighbors marked itself in Step 1 of the algorithm satisfies : $p \geq 1 - e^{-\frac{1}{6}}$.*

Proof: We denote $d(u)$ by d and let $u_1, ..., u_k$ be the neighbors of u with degree smaller than d. Note that $k \geq d/3$.

$$\begin{aligned}
p &= 1 - \Pr[\text{none of } u'\text{s neighbors marked itself}] \\
&\geq 1 - \Pr[\text{none of } u_1, ..., u_k \text{ marked itself}] \\
&= 1 - \prod_{i=1}^{k} \Pr[u_i \text{ didn't mark itself}] \\
&= 1 - \prod_{i=1}^{k} (1 - 1/2d(u_k))
\end{aligned}$$

Since $d(u_i) \leq d$, $p \geq 1 - \prod_{i=1}^{k}(1 - 1/2d) = 1 - (1 - 1/2d)^k \geq 1 - (1 - 1/2d)^{\frac{d}{3}} \geq 1 - e^{-\frac{1}{6}}$. □

Lemma 16.3.2 *Let A be the event that vertex u marked itself in Step 1 of the algorithm, and let B be the event that $u \in MIS$. Then, $Pr[B] \geq Pr[A]/2$.*

Proof: Let N_i be the event that the i^{th} neighbor of u that has higher degree marked itself. Note that $\Pr[N_i] \leq 1/2d$. We have that :

$$\Pr[B] = \Pr[A \cap \text{ no neighbor of higher degree marked itself}]$$

$$= \Pr[A] - \Pr[A \cap \text{ some neighbor of higher degree marked itself}]$$

$$= \Pr[A] - \Pr[A \cap \cup_i N_i] = \Pr[A] - \Pr[\cup_i (A \cap N_i)]$$

$$\geq \Pr[A] - \sum_i \Pr[A \cap N_i] = \Pr[A] - \sum_i \Pr[A] \cdot \Pr[N_i]$$

$$\geq \Pr[A] \cdot (1 - d/2d) = \Pr[A]/2$$

□

Notice that the analysis in the last proof uses only pairwise independence.

Lemma 16.3.3 *If $u \in V$ is good, then the probability that u will be deleted in Step 4 is at least $(1 - e^{-\frac{1}{6}})/2$.*

Proof: Follows directly from the last two lemmas. □

Let a *good edge* be an edge adjacent to a good vertex. Let B denote the set of bad vertices and let G denote the set of good vertices.

Lemma 16.3.4 *The number of good edges in a graph is at least $|E|/2$.*

Proof: Let us direct the graph so that a vertex of low degree points to a vertex of high degree. It follows that,

$$\forall v \in B, d_{in}(v) \geq 2d_{out}(v)$$

Let $E(X, Y)$ denote the number of directed edges from X to Y. Now,

$$2E(B, B) + E(B, G) + E(G, B)$$

$$= \sum_{v \in B} (d_{out}(v) + d_{in}(v))$$

Since $\forall v \in B, d_{in}(v) \geq 2d_{out}(v)$, we get that $\forall v \in B, d_{out}(v) + d_{in}(v) \leq 3d_{in}(v) - 3d_{out}(v)$. Hence,

$$\sum_{v \in B} (d_{out}(v) + d_{in}(v)) \leq 3 \sum_{v \in B} (d_{in}(v) - d_{out}(v))$$

$$
\begin{aligned}
&= 3\sum_{v\in G}(d_{out}(v) - d_{in}(v)) \\
&= 3(E(B,G) - E(G,B)) \\
&\leq 3(E(B,G) + E(G,B))
\end{aligned}
$$

To conclude, we have that,

$$2E(B,B) + E(B,G) + E(G,B) \leq 3(E(B,G) + E(G,B))$$

And therefore,

$$E(B,B) \leq E(B,G) + E(G,B)$$

This proves the lemma, since $E(B,B)$ is the number of edges that are not good. □

It follows from Lemma 16.3.3 that the probability that a good vertex will be deleted is $(1 - e^{-\frac{1}{6}})/2$. It follows from Lemma 16.3.4 that in *any* graph, at least half of the edges are good. Therefore, in each iteration, the expected number of edges deleted is a constant fraction of the total number of edges.

We now proceed to show that the analysis holds when the random choices, i.e., marking the vertices, are pairwise independent. The only proof that uses full independence is the proof of Lemma 16.3.1. We modify the lemma as follows :

Lemma 16.3.5 *Let $v \in B$. Denote by A_i the event that the i^{th} neighbor of v marked itself, and denote by B_i the event that the i^{th} neighbor of v remained marked. Then, $Pr[\cup_i B_i] \geq 1/8 \cdot \min\{\sum_i Pr[A_i], 1\}$.*

Proof: By the Inclusion-Exclusion principle we have,

$$\Pr[\cup_i B_i] \geq \sum_i \Pr[B_i] - \sum_{i<j} \Pr[B_i \cap B_j]$$

Since $\Pr[B_i] \geq 1/2 \cdot \Pr[A_i]$ and the event $B_i \cap B_j \subseteq A_i \cap A_j$, we get

$$
\begin{aligned}
\Pr[\cup_i B_i] &\geq 1/2\sum_i \Pr[A_i] - \sum_{i<j}\Pr[A_i \cap A_j] \\
&= 1/2\sum_i \Pr[A_i] - \sum_{i<j}\Pr[A_i]\Pr[A_j] \\
&\geq 1/2\sum_i \Pr[A_i] - 1/2(\sum_i \Pr[A_i])^2 \\
&= 1/2\sum_i \Pr[A_i](1 - \sum_j \Pr[A_j])
\end{aligned}
$$

Consider two cases :

1. If $\sum_i \Pr[A_i] \leq 1/2$, then : $\Pr[\cup_i Bi] \geq 1/4 \sum_i \Pr[A_i]$.
2. If $\sum_i \Pr[A_i] \geq 1/2$, we choose a subset of the neighbors of v for which $\sum_i \Pr[A_i] = 1/2$ and get : $\Pr[\cup_i B_i] \geq 1/8$.

$\square$

For a good vertex v, we have that $\sum_i \Pr[A_i] \geq d/3 \cdot 1/2d = 1/6$. We conclude that a good vertex v has a marked neighbor after Step 2 with probability at least $1/24$. The rest of the proof of the algorithm remains the same.

16.3.4 Derandomization

In the previous lecture we saw a construction of a sample space of size $O(n^2)$ of n random variables that are pairwise independent. There is a subtle issue when using this construction in our case. There may be an error of at most $1/p$, which is at most $1/2n$, in the probabilities, when generating the random variables. However, this error is negligible as can easily be checked in the analysis of the algorithm.

Clearly, each assignment in the sample space can be checked (independently of the others) by a linear number of processors. Therefore, the algorithm for computing a maximal independent set can be derandomized in NC.

16.3.5 Open Problems

The following distributed model was defined by Linial [Li]. Each vertex of an undirected graph is occupied by a processor. At each time unit a processor may send messages to each of its neighbors, where the message size is unbounded. Any other computation is free, e.g., computations carried out by individual processors. The computation is completely synchronous. The cost of computing a function f which is defined on the graph is the distance that data has to travel (in the graph). Clearly, an upper bound on the cost of any function is the diameter of the graph. The question is which "global" functions can be computed faster than the diameter of the graph.

Two examples of "global" problems that were considered in this model are coloring and computing a maximal independent set. Linial [Li] gave an $O(\log^* n)$ algorithm for coloring a graph using Δ^2 colors, where Δ is the maximum degree. This bound is tight; any improvement in the number of colors would require the cost to be equal to the diameter.

Notice that the parallel randomized algorithm we presented for computing a maximal independent set is a "local" algorithm. At each step, each (active) processor compares its degree with that of its neighbors. The expected number of steps is logarithmic. However, the derandomization process requires "global" information. It is an outstanding open problem to find a deterministic algorithm in this model for computing a maximal independent set such that its cost would be less than the diameter of the graph.

Chapter 17

Small Bias Probability Spaces

In this chapter we show how to construct small probability spaces that behave similarly to larger probability spaces in certain senses. Our approach is different from the one taken in the previous chapter, i.e., reducing the size of the sample space by requiring only limited independence among the random variables (as opposed to full independence). It is based on the equivalence of the following two conditions: (See [CGHFRS] and [Vaz]).

Let $x_1, \ldots, x_n$ be $\{0, 1\}$ Bernoulli random variables.

1. The random variables are independent and for all i, $\text{Prob}[x_i = 0] = \text{Prob}[x_i = 1]$.

2. For every subset $S \subseteq \{1, \ldots, n\}$, it is equally likely that the parity of the subset (i.e., the number of "ones") is either zero or one.

We are going to relax the second condition and construct a probability distribution such that for every subset $S \in \{1, \ldots, n\}$, it is "almost" equiprobable that the parity of the subset is zero or one. To be more precise, we require that for every subset S,

$$\left|\text{Prob}[\sum_{i \in S} x_i = 0] - \text{Prob}[\sum_{i \in S} x_i = 1]\right| \leq \epsilon$$

This quantity was called by Vazirani [Vaz] the *bias* of the subset S.

In this chapter we describe efficient constructions of random variables that have small bias. We first introduce the Fourier transform of Z_n^2.

17.1 Fourier transform of Z_2^n

Let Z_2^n be the group of vectors in $\{0,1\}^n$ where the operation is addition mod 2 (componentwise). The set $f : Z_2^n \to \Re$ of functions forms a vector space where the dimension of the vector space is 2^n.

Definition 17.1.1 *An inner product of two functions $f, g : Z_2^n \to \Re$ is defined to be :*

$$< f, g >= 1/2^n \sum_{x \in Z_2^n} f(x) \cdot g(x)$$

It will be more convenient to define Z_2^n as a group of vectors from $\{-1,1\}^n$ where the addition operation is defined as bitwise multiplication.

Definition 17.1.2 *A character is a homomorphism from the group elements to the roots of unity. Define for each subset $S \subseteq 1 \dots n$ a character χ_S.*

$$\forall x \in Z_2^n,\ \chi_S(x) = \prod_{i \in S} x_i$$

In case that $S = \emptyset$, $\forall x,\ \chi_S(x) = 1$ (the identity character).

The characters serve as a basis for the space of functions from Z_2^n to $\Re$.

Theorem 17.1.1 *The characters constitute an orthonormal base for the vector space of functions $f : Z_2^n \to \Re$.*

Proof:

$$\forall S : < \chi_S, \chi_S >= 1/2^n \sum_x \chi_S^2(x) = (1/2^n) \cdot 2^n = 1$$

$$\forall S, T, S \neq T : < \chi_S, \chi_T >= 1/2^n \sum_x \chi_S(x) \cdot \chi_T(x) = 1/2^n \sum_x \chi_{S \Delta T}(x) = 0$$

where $S \Delta T$ is the symmetric difference of S and T. Now, there are 2^n characters and each vector in the space has 2^n coordinates. Therefore, the characters span the space. $\square$

We are now ready to define the Fourier transform. We describe a function f by projecting it on the basis functions, the characters. The Fourier coefficients of f are, for all subsets $S \subseteq 1, \dots, n$: $\widehat{f}_S =< f, \chi_S >$.

The function f can be reconstructed from its Fourier coefficients as follows.

Theorem 17.1.2 *For all $f : \{-1,1\}^n \to \Re$:*

$$f(x) = \sum_S \widehat{f}_S \chi_S(x)$$

Proof: We fix $x = \overline{x}$.

$$f(\overline{x}) = \sum_S \widehat{f}_S\, \chi_S(\overline{x}) = \sum_S \left(\frac{1}{2^n} \sum_{x \in \{-1,1\}^n} f(x)\chi_S(x) \right) \chi_S(\overline{x}) =$$
$$= \frac{1}{2^n} \sum_{x \in \{-1,1\}^n} \sum_S f(x)\, \chi_S(x)\chi_S(\overline{x}) = \frac{1}{2^n} \sum_S f(\overline{x})\chi_S^2(\overline{x}) + \frac{1}{2^n} \sum_{x \neq \overline{x}} \sum_S f(x)\chi_S(x)\chi_S(\overline{x})$$

Since $\chi_S(\overline{x})^2 = 1$, the first term in the last equality is $f(\overline{x})$. To see that the second term is zero, notice that $\chi_S(x)\chi_S(\overline{x}) = \chi_S(x \cdot \overline{x})$ and that if $x \neq \overline{x}$, then $x \cdot \overline{x} \neq \vec{1}$. This implies that $\chi_S(x \cdot \overline{x}) = 1$ on precisely half the subsets. $\square$

17.1.1 Fourier transform of probability distributions

We use D to denote a probability distribution $D : \{-1,1\}^n \to [0,1]$ such that $\sum_x D(x) = 1$. For any distribution D we have

$$\widehat{D}_\phi = \frac{1}{2^n} \sum_x D(x)\chi_\phi(x) = \frac{1}{2^n} \sum_x D(X) = \frac{1}{2^n}$$

For the uniform distribution, denoted henceforth by U, we have that for every non-empty subset S

$$\widehat{U}_S = \frac{1}{2^n} \sum_x D(x)\chi_S(x) = \frac{1}{2^{2n}} \sum_x \chi_S(x) = 0$$

The converse is also true.

Claim 17.1.1 *If for all $S \neq \phi$, $\widehat{D}_S = 0$, then $D = U$.*

Proof:

$$D(x) = \sum_S \widehat{D}_S \chi_S(x) = \frac{1}{2^n}\chi_\phi(x) = \frac{1}{2^n}$$

$\square$

17.2 Small biased random variables

Let $x_1, \ldots, x_n$ be $\{-1,1\}$ random variables and let D denote as before their joint distribution. We first define the bias of a subset S.

Definition 17.2.1 *The bias of a subset $S \subseteq \{1, \ldots, n\}$ is defined to be,*

$$bias_D(S) = \left| \mathrm{Prob}_D \left[\prod_{i \in S} x_i = -1 \right] - \mathrm{Prob}_D \left[\prod_{i \in S} x_i = 1 \right] \right|$$

The relation between the bias of a subset S and the absolute value of the corresponding Fourier coefficient is given by the following:

Proposition 17.2.1

$$2^n \left|\widehat{D}_S\right| = bias_D(S)$$

Proof:

$$2^n \left|\widehat{D}_S\right| = \left|\sum_x D(x)\chi_S(x)\right| = \left|\sum_x D(x)\prod_{i\in S} x_i\right| =$$

$$= \left|\Pr[\prod_{i\in S} x_i = 1] - \Pr[\prod_{i\in S} x_i = -1]\right| = bias_D(S) \quad \square$$

We now define random variables that have a small bias.

Definition 17.2.2 *The variables $x_1, \ldots, x_n$ are ϵ-biased if for all S, $bias_D(S) \leq \epsilon$.*

Definition 17.2.3 *The variables $x_1, \ldots, x_n$ are k-wise ϵ-biased if for all subsets S such that $|S| \leq k$, $bias_D(S) \leq \epsilon$.*

Definition 17.2.4 *The variation distance between two distributions D_1 and D_2 defined over the same probability space Ω is*

$$\|D_1 - D_2\| = \sum_{\omega\in\Omega} |D_1(\omega) - D_2(\omega)|$$

Let $D(S)$ deonte the distribution D restricted to the subset S.

Definition 17.2.5 *The variables $x_1, x_2, \ldots, x_n$ are defined to be k-wise δ-dependent if for all subsets S such that $|S| \leq k$,*

$$\|U(S) - D(S)\| \leq \delta$$

The next theorem motivates ϵ-bias random variables by linking the variation distance to the sum of the squares of the Fourier coefficients.

Theorem 17.2.1 *Let $x_1, \ldots, x_n$ be $\{-1, 1\}$ random variables and let D be their joint probability distribution. Then,*

$$\|D - U\| \leq 2^n \cdot \left(\sum_{S\subseteq\{1\ldots n\}} \widehat{D}(S)^2\right)^{\frac{1}{2}} = \left(\sum_{S\subseteq\{1\ldots n\}} bias_D^2(S)\right)^{\frac{1}{2}}$$

Proof: The square of the variation distance is,

$$\|D-U\|^2 = \left(\sum_{x\in\{-1,1\}^n} \left|D(x) - \frac{1}{2^n}\right|\right)^2$$

By the Cauchy-Schwarz Inequality * , we get

$$\leq 2^n \cdot \sum_{x\in\{-1,1\}^n} \left(D(x) - \frac{1}{2^n}\right)^2$$

$$= 2^n \cdot \sum_{x\in\{-1,1\}^n} \left(\sum_S \widehat{D}(S)\cdot\chi_S(x) - \frac{1}{2^n}\right)^2$$

Since $\widehat{D}(\emptyset) = \frac{1}{2^n}$ we have,

$$\begin{aligned}
&= 2^n \cdot \sum_{x\in\{-1,1\}^n} \left(\sum_{S\neq\emptyset} \widehat{D}(S)\cdot\chi_S(x)\right)^2 \\
&= 2^n \cdot \sum_{x\in\{-1,1\}^n} \sum_{S_1\neq\emptyset} \sum_{S_2\neq\emptyset} \widehat{D}(S_1)\widehat{D}(S_2)\chi_{S_1}(x)\chi_{S_2}(x) \\
&= 2^n \cdot \sum_{S_1\neq\emptyset} \sum_{S_2\neq\emptyset} \widehat{D}(S_1)\widehat{D}(S_2) \sum_{x\in\{-1,1\}^n} \chi_{S_1}(x)\chi_{S_2}(x) \\
&= 2^{2n} \cdot \sum_{S_1\neq\emptyset} \sum_{S_2\neq\emptyset} \widehat{D}(S_1)\widehat{D}(S_2) < \chi_{S_1}, \chi_{S_2} >
\end{aligned}$$

By orthonormality we have that if $S_1 \neq S_2$, then $< \chi_{S_1}, \chi_{S_2} >= 0$; otherwise, $< \chi_{S_1}, \chi_{S_2} >= 1$. Hence,

$$= 2^{2n} \cdot \sum_{S_1\neq\emptyset} \sum_{S_2\neq\emptyset} \widehat{D}(S_1)\widehat{D}(S_2) < \chi_{S_1}, \chi_{S_2} > = \ 2^{2n} \cdot \sum_S \widehat{D}(S)^2$$

Which implies that,

$$\|D-U\| \leq 2^n \cdot \left(\sum_{S\subseteq\{1\ldots n\}} \widehat{D}(S)^2\right)^{\frac{1}{2}}$$

To complete the proof, recall that the definition of the bias of a subset S is,

$$\begin{aligned}
bias_D(S) &= |\mathrm{Prob_D}[\chi_S(x) = -1] - \mathrm{Prob_D}[\chi_S(x) = 1]| \\
&= \left|\sum_{x\in\{-1,1\}^n} D(x)\chi_S(x)\right| \\
&= 2^n| < D, \chi_S > | = |2^n\widehat{D}(S)|
\end{aligned}$$

*The Cauchy-Schwarz Inequality, proved in Appendix 17.8, is $< f,g >^2 \leq < f,f >< g,g >$. Let $g(x) = 1$. By definition of inner product we get, $(\frac{1}{2^n}\sum f(x)\cdot 1)^2 \leq (\frac{1}{2^n}\sum f(x)^2)(\frac{1}{2^n}\sum 1)$ which implies, $(\sum f(x))^2 \leq 2^n \sum f(x)^2$

and hence,

$$2^n \cdot \left(\sum_{S \subseteq \{1 \ldots n\}} \widehat{D}(S)^2 \right)^{\frac{1}{2}} = \left(\sum_{S \subseteq \{1 \ldots n\}} bias_D^2(S) \right)^{\frac{1}{2}}$$

□

Corollary 17.2.1 *If the random variables $x_1, \ldots, x_n$ are ϵ-biased with respect to a distribution D, then they are also k-wise δ-dependent for $\delta = 2^{k/2} \cdot \epsilon$.*

17.3 An efficient construction

We now provide an efficient deterministic construction of small bias random variables. Let Z_p^* be the multiplicative group modulo p and let g be a generator of Z_p^*. That is, $Z_p^* = \{g^j : j \in \{1, \ldots, p-1\}\}$. In the previous lecture we defined a character to be a homomorphism from a group to the roots of unity. In any cyclic group, (in particular Z_p^*), the character of order r is defined to be:

$$\chi^{(r)}(g^s) \stackrel{def}{=} e^{2\pi i r s/m}$$

Where m is the order of the group. The group Z_p^* has order $p-1$.

Characters have a multiplicative property:

$$\chi^{(r)}(g^s)\, \chi^{(r)}(g^t) = \chi^{(r)}(g^{s+t})$$

We will use the character of order $\frac{p-1}{2}$ which has the following property:

$$\chi^{(\frac{p-1}{2})}(g^s) = e^{\frac{2\pi i (p-1) s}{p-1}} = e^{\pi i s} = \begin{cases} 1 & \text{if } s \text{ is odd} \\ -1 & \text{if } s \text{ is even} \end{cases}$$

Notice that this is precisely the definition of a *quadratic residue*. The following algorithm constructs ϵ-biased random variables.

1. Choose a prime $p > n$.
2. Select, uniformly at random, an x from Z_p^*.
3. For $i = 1, \ldots, n$, define the random variable $x_i = \chi^{(\frac{p-1}{2})}(x + i)$

Remark 17.3.1 *If $x+i=p$ in the above construction, then for all $j \geq i$, $x_j = \chi^{(\frac{p-1}{2})}(x+j+1)$.*

Clearly, each $x \in Z_p^*$ determines an assignment to the random variables. Hence, the size of the sample space is $p-1$. In addition, since the number of quadratic residues is equal to the number of quadratic non-residues,

$$\Pr[X_i = 1] = \Pr[X_i = -1]$$

To prove that the bias of each subset is bounded, we invoke Weil's Theorem.

Theorem 17.3.1 (Weil) *Let p be an odd prime, χ be a character of order $d > 1$, f be a polynomial over $GF[p]$ which is not of order d, ($f(x) \neq c \cdot g(x)^d$ for all g and c). If f has exactly m roots, then*

$$\left| \sum_{x \in Z_p^*} \chi(f(x)) \right| \leq (m-1)\sqrt{p}$$

The proof of this theorem is beyond the scope of this course. We now analyze the bias of the random variables. The following is derived by using the multiplicative property of characters and the fact that x is chosen uniformly at random.

$$\begin{aligned} bias_D(S) &= |\Pr[\textstyle\prod_{i\in S} x_i = 1] - \Pr[\prod_{i\in S} x_i = -1]| \\ &= |\Pr[\chi(\textstyle\prod_{i\in S}(x+i)) = 1] - \Pr[\chi(\prod_{i\in S}(x+i)) = -1]| \\ &= \tfrac{1}{p-1} \left|\textstyle\sum_x \chi(\prod_{i\in S}(x+i))\right| \end{aligned}$$

The expression $\prod_{i\in S}(x+i)$ is a polynomial, $f(x)$, with $|S|$ roots and it cannot be a d^{th} power since it is a product of different monomials. Thus, by Weil's theorem,

$$bias_D(S) = \frac{1}{p-1} \left| \sum_x \chi(\prod_{i\in S}(x+i)) \right| \leq \frac{|S|\sqrt{p}}{p-1} \leq \frac{2n}{\sqrt{p-1}}$$

Let T be a subset of $\{1, \ldots, n\}$, such that $|T| \leq \log n$. We have by Theorem 17.2.1 that

$$\|D_T - U_T\| \leq \sqrt{\sum_{S\subseteq T} bias_D^2(S)} \leq 2^{\frac{\log n}{2}} \cdot \frac{2\log n}{\sqrt{p-1}} = \frac{2\sqrt{n}\log n}{\sqrt{p-1}}$$

If we choose $p \approx n^4$, then for each such T, the distribution restricted to T is "almost" $\log n$-wise independent. However, the sample space Ω is of polynomial size † as opposed to the sample space of $\log n$-wise independent variables.

† $|\Omega| = p = n^4$

17.4 A construction of k-wise ϵ-biased random variables

In this section it will be more convenient to consider random variables over $\{0,1\}^n$, instead of variables over $\{-1,1\}^n$. We modify Definition 17.2.1 for such variables:

Definition 17.4.1 *The bias of a subset $S \subseteq \{1,\ldots,n\}$ for a distribution D is defined to be,*

$$bias_D(S) = \left|\text{Prob}_D\left[\sum_{i\in S} x_i \equiv 0 \pmod 2\right] - \text{Prob}_D\left[\sum_{i\in S} x_i \equiv 1 \pmod 2\right]\right|$$

Recall the construction of k-wise independent random variables in Section 16.1. There, we had vectors $v_1,\ldots,v_n \in \{0,1\}^h$ that were linearly k-wise independent, where $h = \frac{k\log n}{2}$. We chose uniformly at random $r \in \{0,1\}^h$ and defined $x_i =< v_i, r >$, for $i = 1,\ldots,n$. The resulting random variables are k-wise independent.

To construct k-wise ϵ-biased random variables, we use the same construction, only that now we sample r from an ϵ-biased source (instead of a uniformly distributed source). In other words, r is chosen uniformly at random from a sample space of h random variables that are ϵ-bias.

Claim 17.4.1 *The random variables constructed by sampling r from an ϵ-biased source are k-wise ϵ-biased.*

Proof: For every subset S such that $|S| \le k$,

$$\sum_{i\in S} x_i = \sum_{i\in S} < v_i, r >= r\cdot\sum_{i\in S} v_i = R\cdot M_S$$

Where $M_S \stackrel{def}{=} \sum_{i\in S} v_i$. Since the vectors are k-wise independent, $M_S \neq \vec{0}$. Hence,

$$bias(S) = \left|\Pr[\sum_{i\in S} x_i = 1] - \Pr[\sum_{i\in S} x_i = -1]\right| = |\Pr[R\cdot M_S = 1] - \Pr[R\cdot M_S = -1]| \le \epsilon$$

□

17.5 A moment inequality

In this section we formulate a moment inequality for the sum of $\{-1,1\}$ random variables $(x_1,\ldots,x_n)$ that are k-wise ϵ-biased. We denote their sum by $S = \sum_{i=1}^n x_i$. This will be done by bounding their k^{th} moment.

Since $E[S] = 0$,

$$\text{Prob}[|S| \geq \lambda] \leq \frac{E[|S|^k]}{\lambda^k}$$

This is known as the k^{th} moment inequality. (See Theorem 1.3.1).

Let $b_1, \ldots, b_n$ be random variables that have a binomial distribution, i.e., they are independent and take their values from $\{-1, 1\}$ uniformly. The k^{th} moment of the absolute value of their sum will be denoted by B_k, that is $B_k = E\left[|b_1 + \ldots + b_n|^k\right]$. We bound the k^{th} moment of S via B_k. (We assume w.l.o.g. here that k is even). The k^{th} moment of S contains n^k terms, where each term contains at most k variables. More specifically,

$$E[S^k] \;=\; E\left[\sum_{i_1 i_2 \ldots i_k} x_{i_1} x_{i_2} \ldots x_{i_k}\right] \;=\; \sum_{i_1 i_2 \ldots i_k} E\left[x_{i_1} x_{i_2} \ldots x_{i_k}\right]$$

Each term in the above summation is of the form $T_i = x_{i_1}^{p_1} x_{i_2}^{p_2} \ldots x_{i_r}^{p_r}$, such that $\sum_{j=1}^{r} p_j \;=\; k$. If a term T_i contains a variable whose power is odd, then in B_k its expectation is 0. However, in our case, it follows from the definition of k-wise ϵ-biased random variables (Definition 17.2.3), that the expected value of T_i can be at most ϵ. If all the powers in a term T_i are even, then the expected value in both cases is the same and equal to 1. Hence,

Theorem 17.5.1 *Let S and B_k be as above, then*

$$\text{Prob}[|S| \geq \lambda] \leq \frac{B_k + \epsilon \cdot n^k}{\lambda^k}$$

17.6 Lower bounds

In this section we prove lower bounds on the cardinality of ϵ-biased probability spaces. The lower bounds are proved via the intimate relationship between ϵ-bias sample spaces and linear codes. Recall the definition of a linear code (and related notions) in Section 16.1. Let Ω denote a 2ϵ-bias distribution of n random variables $x_1, \ldots, x_n$. We construct the following $n \times m$ matrix G: each column corresponds to a point in Ω, i.e., each column is a possible assignment to the random variables. We can view G as a generating matrix of a linear code C.

Proposition 17.6.1 *Let $c \in C$ be any codeword and let* $\mathrm{wt}(c)$ *denote its weight. Then,*

$$\frac{1}{2} - \epsilon \leq \frac{\mathrm{wt}(c)}{m} \leq \frac{1}{2} + \epsilon$$

Proof: Let v be the information word corresponding to c, i.e., $v \cdot G = c$. We can view v as a characteristic vector of a subset $S \subseteq 1, \ldots, n$. From the definition of a 2ϵ-bias probability space we have :

$$\left| \text{Prob}\left[\sum_{i \in S} x_i = 0 \pmod 2\right] - \text{Prob}\left[\sum_{i \in S} x_i = 1 \pmod 2\right]\right| \leq 2\epsilon$$

Hence, the number of "1"s in $v \cdot G$ is at least $m(1/2 - \epsilon)$ and at most $m(1/2 + \epsilon)$. □

Recall again from Section 16.1 that for a linear code, the minimum weight of a codeword (that is not the zero word) is also the minimum distance between two codewords. Therefore, the known bounds in the theory of error correcting codes can be used to derive lower bounds on the cardinality of small bias sample spaces.

We proceed formally. Let the function $m(n, \epsilon)$ denote the minimum m such that there exists a sample space of size m corresponding to n $\{0, 1\}$ random variables that are 2ϵ-bias, or, in other words, the probability that any non-trivial combination of the random variables is zero, is between $1/2 - \epsilon$ and $1/2 + \epsilon$.

It is shown in [AGHP] (and also in [ABNNR]) that the following bound holds:

$$m(n, \epsilon) \geq \Omega\left(\min\left\{\frac{n}{\epsilon^2 \log(1/\epsilon)}, 2^n\right\}\right)$$

It follows that our upper bound constructions are pretty tight with this bound.

17.7 Further results

We have seen so far how to construct an ϵ-bias probability space with respect to the uniform distribution. An interesting open problem is how to construct an ϵ-bias distribution with respect to a non uniform $\{0, 1\}$ distribution. More formally, let D be some arbitrary distribution on $\{0, 1\}$ random variables $x_1, \ldots x_n$ and let $\widehat{D}_S$ (for all subsets S) denote its Fourier coefficients. We wish to construct a distribution D' such that,

$$\forall S \subseteq \{1 \ldots n\}, \ |\widehat{D}'_S - \widehat{D}_S| \leq \frac{\epsilon}{2^n}$$

$$|D'| = \text{poly}(n, 1/\epsilon)$$

Azar, Motwani and Naor [AMN] showed (using the probabilistic method) that for every distribution D, there exists a distribution D' that satisfies the above conditions.

However, it is an open problem to obtain a deterministic efficient construction of D' even in the case where the random variables are independent and each random variable x_i is defined as follows.

$$x_i = \begin{cases} 0 & \text{with probability} \quad p \\ 1 & \text{with probability} \quad 1-p \end{cases}$$

In fact, in this case, the Fourier coefficients of D have a simple description and coefficients corresponding to equal cardinality subsets are equal. It is also easy to generalize Theorem 17.2.1 to arbitrary distributions and hence, by Corollary 17.2.1, an ϵ-bias distribution is also k-wise δ-dependent (for arbitrary distributions) for $\delta = 2^{\frac{k}{2}} \cdot \epsilon$. Notice, however, that the converse is not true: a k-wise δ-dependent distribution is not necessarily a k-wise ϵ-bias distribution.

In the following we survey the results related to constructions of small bias probability spaces. The first construction is due to Naor and Naor [NN] who gave an efficient construction of an ϵ-bias probability space for random variables that are uniform on $\{0,1\}$. Simpler constructions of the same probability space were given by Alon, Goldreich, Hastad and Peralta [AGHP]. In particular, the construction based on quadratic residues is taken from [AGHP]. Azar, Motwani and Naor [AMN] constructed a probability space for ϵ-bias random variables over any integral domain $[0 \ldots d-1]$, $d \geq 2$. A recent result [EGLNV], is a polynomial construction for k-wise, δ-dependent general random variables over any integral domain, where $k = \log n$, $\delta = \frac{1}{\text{poly}(n)}$.

17.8 Appendix: the Cauchy-Schwarz Inequality

Theorem 17.8.1 (Cauchy-Schwarz Inequality)

$$< f, g >^2 \leq < f, f >< g, g >$$

Proof: We use the following easy to verify properties of inner products:

Claim 17.8.1 *For any real functions f, g, h over $\{-1, 1\}^n$ and any constant $a \in \mathcal{R}$*

$$\begin{aligned} < f + g, h > &= < f, h > + < g + h > && (17.1)\\ < f, g + h > &= < f, g > + < f, g > && (17.2)\\ < af, g > &= a < f, g > && (17.3)\\ < f, g > &= < g, f > && (17.4)\\ < f, f > &\geq 0 && (17.5) \end{aligned}$$

For any real functions f, g, and a constant $\lambda \in \mathcal{R}$ we have

$$\begin{aligned} 0 &\leq < \lambda f - g, \lambda f - g > \\ &= < \lambda f, \lambda f - g > - < g, \lambda f - g > \\ &= \lambda^2 < f, f > -2\lambda < f, g > + < g, g > \end{aligned}$$

Substituting $\lambda = \frac{<f,g>}{<f,f>}$ we get,

$$\begin{aligned} 0 &\leq \frac{< f, g >^2}{< f, f >^2} < f, f > -2\frac{< f, g >^2}{< f, f >} + < g, g > \\ &= < g, g > - \frac{< f, g >^2}{< f, f >} \end{aligned}$$

Rearranging the last inequality, we get

$$< f, g >^2 \leq < f, f >< g, g >$$

$\square$

Chapter 18

Applications of Small Bias Probability Spaces

In this chapter we present four different applications of small bias probability spaces.

18.1 Set balancing revisited

Random variables that are ϵ-biased are useful for derandomization. We demonstrate this with the problem of set balancing introduced earlier.

A collection of subsets $\mathcal{S} = \{S_1, S_2, \cdots S_n\}$ defined over a base set $B = \{b_1, b_2, \cdots b_n\}$ is given such that the cardinality of each subset is δ. The output is a $\{-1, 1\}$ coloring of B into two sets. Let the 2-coloring of B be denoted by $\mathbf{x} = (x_1, x_2, \cdots x_n)$. The discrepancy of a subset S_i, $\Delta(S_i, \mathbf{x})$, with respect to $\mathbf{x}$, is defined as $\sum_{j \in S_i} x_j$. The discrepancy of the set family $\mathcal{S}$ is defined as,

$$\Delta(\mathcal{S}, \mathbf{x}) = \max_{S_i} \ \Delta(S_i, \mathbf{x})$$

Definition 18.1.1 *A 2-coloring $\mathbf{x}$ is τ-good for a set family $\mathcal{S}$ if there exists τ, $0 < \tau < \frac{1}{2}$, such that,*

$$\Delta(\mathcal{S}, \mathbf{x}) \ \leq \ \delta^{0.5+\tau}\sqrt{\log n}$$

We want to show that if we choose a random coloring $x_1, \ldots, x_n$ such that the random variables are k-wise ϵ-biased, the probability that the coloring is τ-good is non-zero. Let T_i deonte the sum of the random variables in the subset S_i.

If the variables $x_1, \ldots, x_n$ are k-wise independent, then we have the following lemma proved by [BR, MNN].

Lemma 18.1.1 *Let $k = \frac{a \log n}{\log \delta}$ where $a > \frac{1}{\epsilon}$. Let $x_1, \ldots, x_n$ be k-wise independent random variables over $\{-1, 1\}$ and let $disc \stackrel{def}{=} \delta^{0.5+\epsilon}$. Then*

$$\Pr[|T_i| > disc] < \frac{1}{\delta^{\tau k}} < \frac{1}{n}$$

Proof: Homework assignment. Follows from the k^{th} moment inequality. □

We want to bound via Theorem 17.5.1 the increase in the k^{th} moment when k-wise ϵ-biased random varibles are used. The above lemma implies, for a subset S_i, that

$$\begin{aligned}\Pr[|S_i| \geq \delta^{\frac{1}{2}+\tau}\sqrt{\log n}] &\leq \frac{E[S_i^k]}{(\delta^{0.5+\tau}\sqrt{\log n})^k} \\ &\leq \frac{B_k}{(\delta^{\frac{1}{2}+\tau}\sqrt{\log n})^k} + \frac{\epsilon\delta^k}{(\delta^{0.5+\tau}\sqrt{\log n})^k} \\ &< \frac{1}{\delta^{\tau k}} + \frac{E[T_i^k]}{(\delta^{0.5+\tau}\sqrt{\log n})^k}\end{aligned}$$

Hence, ϵ and k must be chosen such that

$$\frac{1}{\delta^{\tau k}} + \frac{\epsilon\delta^k}{(\delta^{0.5+\tau}\sqrt{\log n})^k} < \frac{1}{n}$$

We choose $k = \frac{\log 2n}{\tau \log \delta}$ and the above inequality holds if

$$\epsilon < \frac{1}{2n^{1+\frac{1}{\tau}}}$$

Hence, the sample space will be at most of cardinality $n^{O\left(\frac{1}{\tau}\right)}$ and if τ is a constant, an exhaustive search of the sample space can be conducted. Both the construction and the search can be done in NC^1.

18.2 Testing of VLSI circuits

Assume we are given a VLSI circuit such that each gate depends on at most k (out of n) of the inputs. A problem that has received much attention in the fault diagnostic literature is that of generating a small set of vectors $T \subset \{0, 1\}^n$ such that for any k-subset of indices $S = \{i_1, i_2, \ldots i_k\}$, the projection of T on the indices S contains all possible 2^k configurations. See [SB] for a bibliography on the problem. Such a set of vectors is called (n, k)-universal. The motivation for this problem is that such a test set allows exhaustive testing of a circuit where each component relies on at most k inputs. The best known bounds for constructing (n,k) universal sets are given in [SB] and [Al1].

The trivial solution for this problem is ofcourse to generate all 2^n possibilities. A less trivial solution is to choose as the (n,k) universal set the sample space of n random variables that are k-wise independent. As we saw in a previous lecture, this construction yields $n^{\frac{k}{2}}$ vectors. However, if $k = O(\log n)$, the cardinality of the (n,k) universal set would be super polynomial. There is redundancy in this solution since for every k subset, every combination appears an equal number of times (while all we need is that every combination will appear at least once). We can improve over this construction by using k-wise δ-dependent probability spaces :

Proposition 18.2.1 *If Ω is a k-wise δ-dependent probability space for $\delta \leq 2^{-k}$, then Ω is also a (n,k)-universal set.*

Proof: If for a k-subset $i_1, i_2, \ldots i_k$, there is a $\{0,1\}^k$ configuration which has probability zero in Ω, then the distance from the uniform distribution of $x_{i_1}, x_{i_2}, \ldots x_{i_k}$ is at least $2^{-k} + 2^{-k} = 2^{-(k-1)} > \delta$, where the first summand is due to the missing of the configuration, and the second summand is due to the fact that the weight of the missing configuration is distributed among the other configurations. $\square$

By using the construction of k-wise ϵ-bias random variables given in Section 17.4, we can construct a probability space Ω which is k-wise 2^{-k}-dependent such that the cardinality of Ω is $\log n \cdot 2^{O(k)}$. This provides the first explicit construction of an (n,k)-universal set of polynomial size for k which is $\Theta(\log n)$. It is also better than the best explicit construction given in [SB] for $k << n$, and matches the lower bound given there up to constant factors. Alon [Al1] has shown an explicit construction of size $\log n \cdot 2^{O(k^2)}$ which is optimal for constant k.

It is interesting to note that the construction for a (k,n) universal set can be phrased in coding theory terminology. Let G be the generating matrix of a binary $[n', k', d]$ linear code for which

$$\frac{d}{n'} = \frac{1}{2} - \frac{1}{2^{2k}}$$

and let H be the parity check matrix of a binary $[n, n', k]$ linear code. The (n,k) universal set consists of the rows of the matrix $G \cdot H$.

18.3 Reducing the number of random bits - matrix multiplication verification

In this section we show how to reduce the number of random bits needed in a randomized algorithm for verifying matrix multiplication.

In the matrix multiplication verification problem we are given three $n \times n$ matrices A, B and C and we want to verify that $A \cdot B = C$. This problem was introduced by Freivalds [Fr] who also gave an $O(n^2)$ randomized algorithm that required $O(n)$ random bits. Freivalds's algorithm is as follows :

- Choose a random vector $\vec{r}$ such that each entry in $\vec{r}$ is picked uniformly at random to be zero or some fixed non-zero element.
- Test whether $\vec{r} \cdot A \cdot B - \vec{r} \cdot C = 0$.

The complexity of applying this procedure is that of multiplying a matrix by a vector which is obviously $O(n^2)$.

Theorem 18.3.1 *Suppose that $\vec{r}$ is a vector chosen in the manner described above. Then with probability at least $\frac{1}{2}$, if $A \cdot B \neq C$, then also $\vec{r} \cdot A \cdot B \neq \vec{r} \cdot C$.*

Proof: Let $\vec{v}$ be a non-zero column vector of $A \cdot B - C$ and let j denote the number of its non-zero entries. Let us assume w.l.o.g. that $v_1 = 1$. If we want $\vec{v} \cdot \vec{r} = 0$, every choice of $r_2, \ldots, r_n$ determines the value of r_1. Therefore, the number of different vectors $\vec{r}$ such that $\vec{v} \cdot \vec{r} = 0$ is at most $2^{j-1} \cdot 2^{n-j} = 2^{n-1}$, whereas the number of distinct choices for $\vec{r}$ is 2^n. Hence, the probability that $\vec{v} \cdot \vec{r} = 0$ is at most $\frac{1}{2}$. □

The number of random bits needed for this algorithm is $O(n)$. The proof of the next lemma follows from the definition of ϵ-bias random variables.

Lemma 18.3.1 *Let $\vec{v}$ be a non-zero vector, Ω an ϵ-bias probability space and $\vec{r}$ a column vector chosen uniformly from Ω. Then,* $\Pr[\vec{v} \cdot \vec{r} \neq 0] \geq \frac{1}{2} - \frac{\epsilon}{2}$.

It follows that if we choose a random vector $\vec{r}$ from an ϵ-bias probability space, where ϵ is fixed, $\epsilon < 1$, then the probability that $\vec{v} \cdot \vec{r} = 0$ is a constant. Also, the number of random bits needed to choose a random vector from the above probability space is $O(\log n)$. Therefore, the complexity of the algorithm remains $O(n^2)$ and the number of random bits is reduced to be logarithmic in n.

18.4 Constructing Ramsey graphs

A graph that does not contain a clique or an independent set of size k is called a *k-Ramsey graph*. Erdös [Er] proved in 1947 that most of the graphs on N vertices do not contain a clique or an independent set of size $2\log N$. We consider here the problem of constructing Ramsey graphs.

We state the problem as follows : given a number t, what is the largest graph (as a function of t) that can be explicitly constructed which does not contain a clique or independent set of size t. Let N denote the number of vertices in this graph. Ideally, we would like to find a graph of exponential size. Erdös posed as a challenge the problem of constructing a graph whose size is superpolynomial in t. The challenge was answered by Frankl [Fr] and Frankl and Wilson [FW] who showed an explicit way of constructing graphs where $N = t^{\frac{\log t}{\log\log t}}$. We show a simpler construction, though not as good as [FW], by Moni Naor [Na] which is of size $N = t^{\frac{c\sqrt{\log\log t}}{\log\log\log t}}$ for some fixed c.

When discussing constructive methods of generating combinatorial objects with certain properties, we make a distinction between methods that are *merely constructive* versus those that are *explicit*. By merely constructive we mean that there exists an algorithm that given the parameters of the problem, generates the object in time that is bounded by a polynomial in the input size. A good example is Ajtai's [Aj] construction, where a polynomial time algorithm (in the number of vertices) for constructing a 3-regular expander is given.

By a (locally) explicit construction we mean that given the parameters of the problem and a specific "location" in the object, e.g., an edge in a graph, there is an algorithm that generates the object in that location where the running time of the algorithm is bounded by a polynomial in the number of bits needed to describe the location. A good example is the expander construction of [LPS] where the running time of the algorithm for testing whether two vertices are adjacent is bounded by a polynomial in the logarithm of n, the number of vertices.

The constructions that we shall henceforth exhibit will be explicit in the strong sense, i.e., locally explicit.

The construction of Ramsey graphs presented here is based on combining many graphs, most of which are known to be good, without testing which are the bad ones. The fact that most of them are good compensates for the bad ones.

This is inspired by Justesen's construction of good error correcting codes. It was known that good binary codes can be obtained by concatenating a good code over a large alphabet (which can be explicitly constructed) with good codes that map the large alphabet to the small one. The existence of this code was only proved non-constructively. Justesen suggested using a collection of codes, most of which were known to be good, instead of one good code.

A well known conjecture in this area is that the Paley graph is a Ramsey graph. The Paley graph is defined for every prime p which is congruent to 1 mod 4. There is an edge between nodes i and j iff $i-j$ is a quadratic residue mod p. An interesting consequence of the construction here is that a closely related graph can be shown to be a (weak) Ramsey graph.

The basic technique is due to Abbot [Ab]. Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be graphs on n_1 and n_2 nodes respectively that do not contain a clique or independent set of size k_1 and k_2 respectively. We define a product graph $G = G_1 \times G_2$ as follows: assume the vertices of G_1 (G_2) are labeled $u_1, v_1, \ldots$ ($u_2, v_2, \ldots$). The vertices of G are all the pairs (x, y), where $x \in V_1$ and $y \in V_2$. The edge set of G is defined as follows. Two vertices, (v_1, v_2) and (u_1, u_2), are neighbors in the product graph, if u_1 is different from v_1 and they are adjacent in G_1, or if $u_1 = v_1$, and u_2 is adjacent to v_2 in G_2. An example of the product operation appears in the next page. It is clear that the number of vertices in $G_1 \times G_2$ is $n_1 \cdot n_2$. We claim that $G_1 \times G_2$ does not contain a clique or independent set of size $k_1 \cdot k_2$. To see that, we view the product graph as composed of "big" vertices, where each represents a copy of G_2. Also, the members of a "big" vertex v_1 are connected to all the members of all the "big" vertices which are connected to v_1 in G_1. Let C be a clique, $C \subseteq G_1 \times G_2$. At most k_1 "big" vertices belong to C, and in each "big" vertex, at most k_2 vertices belong to C. Hence, $|C| \leq k_1 \cdot k_2$. A similar analysis can be made for independent sets.

The above observation can be generalized as follows: let $G_1, G_2, \ldots G_m$ be graphs of size $n_1, n_2, \ldots n_m$ such that G_i does not contain a clique or independent set of size k_i. Then, the graph G which is the product graph of $G_1, G_2, \ldots G_m$ is of size $\prod_{i=1}^{m} n_i$ and does not contain a clique or independent set of size $\prod_{i=1}^{m} k_i$.

Suppose we are given a probability space of size m where every point in the probability space is a graph on n vertices. Assume that the probability that a graph is not a k-Ramsey graph is at most α. Let G be the product graph of all the graphs in the probability space. Clearly, $|G| = n^m$. Also, G has no clique or independent set of size $t = k^{(1-\alpha)\cdot m} \cdot n^{\alpha \cdot m}$. This is due to the fact that there are at least $(1-\alpha) \cdot m$ k-Ramsey graphs, and the rest of the

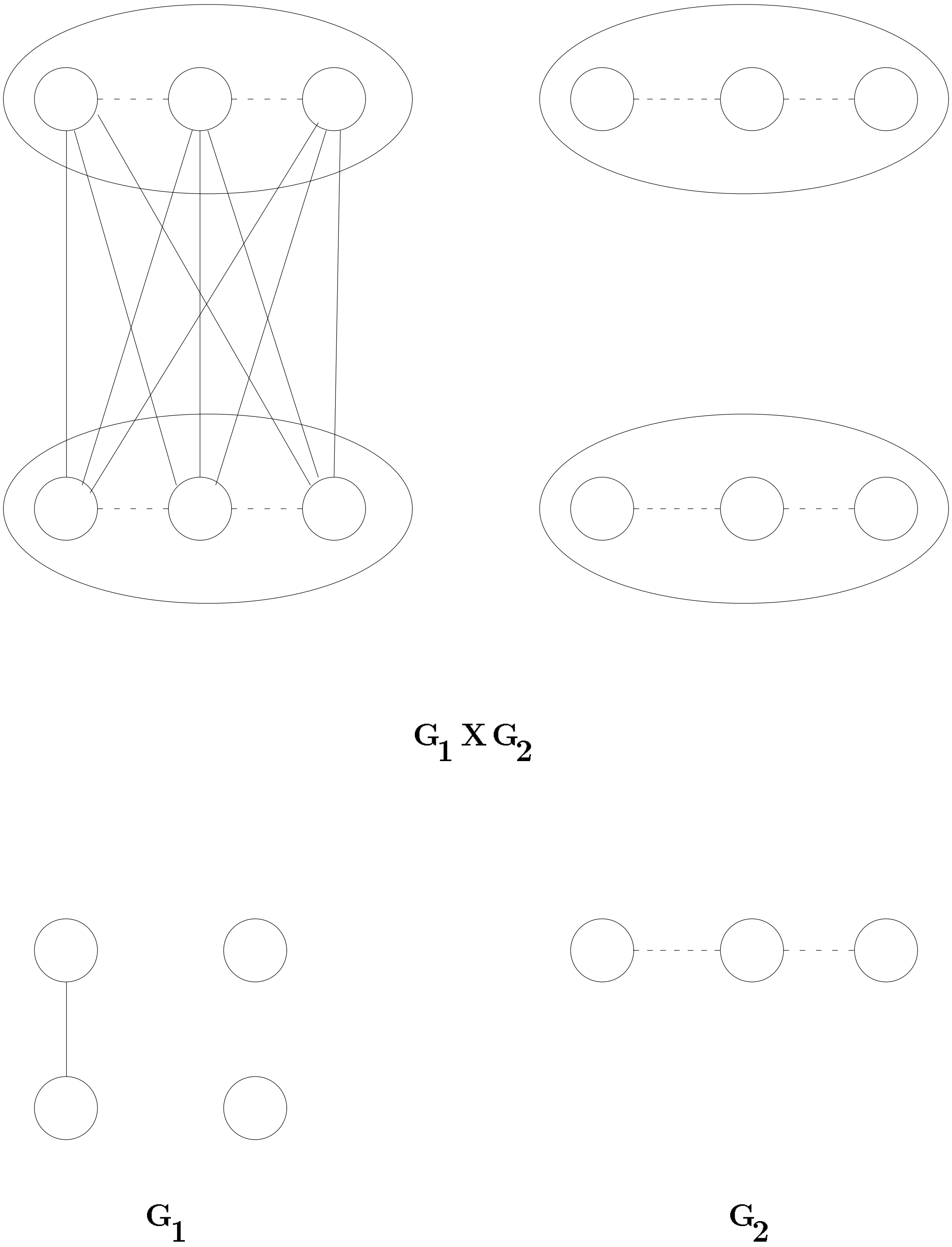


Figure 18.1: An example of the product operation

graphs contribute at most a factor of $n^{\alpha\cdot m}$ vertices to a clique or independent set. Also, if $\alpha \leq \frac{1}{\log n}$, we have that

$$t \leq k^{(1-\alpha)\cdot m} \cdot n^{\alpha\cdot m} < k^m \cdot n^{\alpha\cdot m} < k^m \cdot n^{\frac{1}{\log n}\cdot m} = (2\cdot k)^m$$

As an example to the technique describe above, consider the probability space $G_{n,\frac{1}{2}}$. Erdös proved that for $k = 2\log n$, $\alpha < \frac{1}{\log n}$. Therefore, the product graph of all the graphs in $G_{n,\frac{1}{2}}$ gives us $N = n^{2^{\binom{n}{2}}}$ vertices and $t = (2\log n)^{2^{\binom{n}{2}}}$ vertices. Therefore, $N \geq t^{\frac{\log\log\log t}{\log\log\log\log t}}$.

To improve the above result, we use l-wise ϵ-bias probability spaces. We choose the edges of a graph from a $\binom{k}{2}$-wise ϵ-bias probability space on $\binom{n}{2}$ random variables. The probability that $\binom{k}{2}$ variables are all zero or all one is $2^{-\binom{k}{2}+1} + 2^{\frac{\binom{k}{2}}{2}} \cdot \epsilon$. The first summand is the same as for $\binom{k}{2}$-wise independent random variables, and the second summand is added since the variation distance of the distribution of any $\binom{k}{2}$ random variables from the uniform distribution is at most $2^{\frac{\binom{k}{2}}{2}} \cdot \epsilon$.

Let G be a graph that is chosen from this probability space. The probability that G is not a k-Ramsey graph is bounded above by $\binom{n}{k} \cdot (2^{-\binom{k}{2}+1} + 2^{\frac{\binom{k}{2}}{2}} \cdot \epsilon)$. Taking k to be $2 \cdot \log n$ and ϵ to be 2^{-k^2} we get that this probability is less than $\frac{1}{\log n}$. The construction seen in the previous lecture gives us a probability space of size $(n/\epsilon)^{c'}$ for some fixed c'. After calculations we get that $N \geq t^{\frac{c\sqrt{\log\log t}}{\log\log\log t}}$ for some fixed c.

It remains to show that this construction is indeed locally explicit. We construct a $\binom{k}{2}$-wise ϵ-bias probability space based on quadratic characters. Let p be a prime, $p \geq (k/\epsilon)^2$ and $p \geq \binom{n}{2}$. The random variable x_j at point i is the quadratic character of $i + j \bmod p$, i.e if $i + j$ is a quadratic residue modulo p, then x_j is 1 and it is 0 otherwise. Since the random variables we are interested in are edges, for nodes a and b such that $1 \leq b < a \leq n$ we associate $x_{\frac{a(a-1)}{2}+b}$ with the edge (a, b).

We now look at the product graph $G = G_1 \times G_2 \times \ldots \times G_{p-1}$. Every vertex in G is a choice of $p - 1$ vertices, $(a_1, \ldots, a_{p-1})$ where $a_i \in V_i$. To determine whether there is an edge between two nodes $a = (a_1, a_2, \ldots a_{p-1})$ and $b = (b_1, b_2, \ldots b_{p-1})$, let i be the first index where $a_i \neq b_i$ (i.e. $a_1, a_2, \ldots a_{i-1} = b_1, b_2, \ldots b_{i-1}$). In case $b_i < a_i$, the quadratic residuosity modulo p of $i + \frac{a_i(a_i-1)}{2} + b_i$ determines whether there is an edge (a, b), and in case $a_i < b_i$ it is the quadratic residuosity of $i + \frac{b_i(b_i-1)}{2} + a_i$ that determines whether the edge (a, b) exists.

Chapter 19

Homework

1. Let X be a random variable such that $E[X] = 0$ and $\mathrm{VAR}[X] = \sigma^2$. Prove the following (stronger) version of Chebyshev's Inequality.

$$\Pr[X > \lambda] \leq \frac{\sigma^2}{\sigma^2 + \lambda^2}$$

(Hint: choose a suitable function for the generalized inequality shown in class).

2. Let $X_1, \ldots, X_n$ be independent random variables that are geometrically distributed with expectation equal to 2, and let $S = \sum_{i=1}^{n} X_i$. Use Chernoff bounds and generating functions to give an upper bound on the probability of the tail of S, i.e.,

$$\Pr[S > (1+\delta)2n]$$

for any $\delta > 0$.

3. Let X be a random variable such that $\forall n \geq 0$, $E[X^n] < \infty$. Recall the following two inequalities on the tail of X:

$$\text{Chernoff's inequality :} \quad \Pr[|X| \geq \lambda] \leq \frac{E[e^{t|X|}]}{e^{t\lambda}}$$

$$\text{The } k^{th} \text{ moment inequality :} \quad \Pr[|X| \geq \lambda] \leq \frac{E[|X|^k]}{\lambda^k}$$

Prove, using the probabilistic method, that for every λ, there exists a k such that the k^{th} moment inequality gives a tighter bound than Chernoff's inequality (even if t is optimally

chosen).
(Hint: check the Taylor expansion of $E[e^{t|X|}]$).

4. For a graph G and a subset T (for Top) of the vertices, call an edge "crossing" if it has exactly one vertex in T and let $N(G;T)$ denote the number of such edges. Prove, for appropriate $\alpha = \alpha(n)$: there exists a graph G of $2n$ vertices so that for every set T of n vertices

$$\left| N(G;T) - \frac{n^2}{2} \right| < \alpha$$

What is the smallest α that you can get?
Hint: use the $G_{2n,\frac{1}{2}}$ family of random graphs.

5. Call a $0-1$ matrix A *ninefree* if there is no 3×3 submatrix with all entries one. (Rows and columns need not be consecutive). Let $f(n)$ denote the maximal number of ones in an $n \times n$ ninefree matrix A. Find a lower bound for $f(n)$, i.e., show, for α as large as possible, that there exists a ninefree $n \times n$ A with at least α ones.
Hint: use the deletion method.

6. Let $A_1, \ldots, A_n$ be events in some probability space Ω and let G be their dependency graph as defined in Chapter 6. Suppose that:

1. The probability of each event A_i, $1 \leq i \leq n$ is at most p.
2. The degree of each vertex of G is upper bounded by d.
3. It holds that $4pd \leq 1$.
4. For all $S \subseteq \{1 \ldots n\}$, and for all $i \in \{1 \ldots m\}$, if i and all its neighbors in G are not in S then

$$\Pr[A_i | \bigcap_{j \in S} \overline{A_j}] \leq \Pr[A_i]$$

Does the event $\bigcap_{i=1}^{m} \overline{A_i}$ have positive probability?

7. Let $f(k)$ be the smallest n such that if all subsets of $\{1 \ldots n\}$ with cardinality k are colored by two colors, then there exists a set $T \subseteq \{1 \ldots n\}$ of cardinality $k+1$ for which all subsets of cardinality k have the same color. We stress that subsets, and not elements, are

colored. Give lower bounds on $f(k)$ using the Local Lemma and the (basic) probabilistic method. Which bound is better?

8. (a) Prove that a CNF formula in which each clause contains 100 literals and each literal appears in exactly 100 clauses is satisfiable.
(b) Give a randomized algorithm for finding a satisfying assignment for a CNF formula in which each clause contains k literals and each literal appears in at most $2^{k/100}$ clauses.

9. The following algorithm finds a satisfying assignment for a 2-CNF formula:

1. Begin by assigning a random assignment (independently and uniformly) to each of the variables.
2. While there exists a clause $C_j = (l_{i_1} \vee l_{i_2})$ which is not satisfied by the current assignment:
    - Choose one of the literals in C_j and flip its assignment (so that C_j is now satisfied by the new assignment).

Assume that there exists a satisfying assignment. What is the expected number of times that the algorithm executes the "while" statement until it finds a satisfying assignment?

10. Suppose that we perform a random walk on a ring, starting from an arbitrary vertex, and proceeding until all vertices are visited. Prove that each vertex has an equal probability of being the last vertex visited, independent of the choice for the start vertex of the random walk.

11. Let G be a graph in which each edge (i, j) has a real positive weight C_{ij}, which is defined to be the cost of traversing the edge. The weights satisfy $C_{ij} = C_{ji}$, and $C_{ii} = 0$. Let $N(i)$ denote the set of vertices that are adjacent to vertex i. We define a random walk on this graph with the following probability of moving from vertex i to vertex j:

$$p_{ij} = \frac{1/C_{ij}}{\sum_{k \in N(i)} 1/C_{ik}}$$

Denote by S_{ij} the expected cost of a random walk that starts in i, visits j and returns to i (the first time). Denote by R_{ij} the effective resistance in the graph between vertex i and vertex j (when the resistance of each edge is set to C_{ij}). Show that

$$S_{ij} = 2mR_{ij}$$

12. Let $G = (V, E)$ be a graph such that $|E| = m$ and $|V| = n$. An assignment of weights to the edges is called *good* if for every vertex $r \in V$ it satisfies the following requirements:

- For every vertex $v \in V$, the shortest path from r to v is unique.
- For any two vertices $u, v \in V$, the length of the shortest path from r to u is different from the length of the shortest path from r to v.

Let the weights assigned be integers from the range $[0..R]$. Use the probabilistic method to bound from above the value of R for which a good assignment of weights exists. (Try to minimize the value of R).

13 (a) Let k be the size of the maximum matching in a graph $G = (V, E)$, where $|E| = m$ and $|V| = n$. Prove that the rank of the Tutte matrix of G is $2k$. (Use Frobenius's Theorem: let A be an $n \times n$ matrix for which $A_{i,j} = -A_{j,i}$, and let α, β be subsets of $\{1, 2, \ldots, n\}$ such that $|\alpha| = |\beta| = \text{rank}(A)$. Then, $|A_{\alpha\alpha}| \cdot |A_{\beta\beta}| = (-1)^{|\alpha|} \cdot |A_{\alpha\beta}|^2$.
(b) Devise randomized algorithms for computing the size of the maximum matching in a graph and for finding the maximum matching. (Hint: first solve the problem of computing the maximum/minimum weight perfect matching).
(c) Suggest a more efficient implementation of testing whether the Tutte matrix is identically zero. (By reducing the bit complexity of the test).

14. Prove in a different way than Valiant that there exists a monotone formula for computing the MAJORITY function of depth $O(\log n)$ in which the number of gates is bounded by a polynomial in n. (The MAJORITY function is defined to be "1" if the number of inputs whose value is "1" is strictly greater than $n/2$).
Hint: construct a ternary tree in which every vertex in the tree is a MAJORITY gate and the connections between consecutive levels of the tree are randomly chosen.

15. What is the threshold function in $G_{n,p}$ for subgraph H in Figure 19.1.

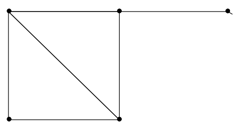


Figure 19.1: Subgraph H

16. The following questions refer to the paper "The probabilistic method yields deterministic parallel algorithms" by R. Motwani, J. Naor & M. Naor [MNN].
Let $C = C_1, C_2, \ldots, C_n$ be k-wise independent random variables generated as follows. Let H be an $h \times n$ matrix where $h = k \log n/2$ and the rows of H are linearly k-wise independent. Let $C = H \cdot R$ where R is a random vector chosen uniformly from $\{0,1\}^h$.

(a) Prove Lemma 3.2:
Let T be a random variable defined by $T = C_1 \cdot C_2 \cdots C_t$, where $t \leq k$.
Prove that $E[T|(r_1, \ldots, r_m) = (d_1, \ldots, d_m)]$ can be computed in NC, for every vector $(d_1, \ldots, d_m) \in \{0,1\}^m$, where $m \leq t$.
Assume that for a set of vectors, the linear dimension of the set and a basis (which is a subset) can be computed in NC.

(b) Prove lemma 4.1:
The discrepancy for which we are aiming is $disc = \delta^{0.5+\epsilon}$. Use the k^{th} moment inequality to prove that there is a positive probability for an ϵ-good coloring, i.e.,

$$\text{Prob}[|X_i - E[X_i]| > disc] < \frac{1}{n}$$

.
Hints: let $k = \frac{\log n}{\epsilon \log \delta}$ and assume that $\epsilon > \frac{1}{\log \delta}$.

(c) Explain the usage of the above two lemmas for the algorithm in Section 4.2.

(d) Explain the considerations in choosing the value of k. What happens when k gets bigger or smaller ?

Bibliography

[Ab] H. L. Abbot, "A note on Ramsey's theorem", Canadian Math. Bull., Vol. 15, 1972, pp. 9-10.

[Ad] L. Adleman, "Two theorems on random polynomial time", 19th Annual Symposium on Foundations of Computer Science, 1978, pp. 75-83.

[Aj] M. Ajtai, "Recursive construction for 3-regular expanders", Proceedings of the 28th IEEE Symposium on Foundations of Computer Science, 1987, pp. 295-304.

[AKS] M. Ajtai, J. Komlos and E. Szemeredi, "Deterministic simulation in LOGSPACE," Proceedings of the 19th ACM Annual Symposium on Theory of Computing, 1987, pp. 132–140.

[Al1] N. Alon, "Explicit constructions of exponential sized families of k-independent sets", Discrete Math, 58, 1986, pp. 191-193.

[Al2] N. Alon, "A parallel algorithmic version of the local lemma", Random Structures and Algorithms, Vol. 2, 1991, pp. 367-378.

[ABI] N. Alon, L. Babai and A. Itai, "A fast and simple randomized algorithm for the maximal independent set problem", J. of Algorithms, Vol. 7, 1986, pp. 567-583.

[ABNNR] N. Alon, J. Bruck, J. Naor, M. Naor and R. Roth, "Construction of Asymptotically Good Low-Rate Error-Correcting Codes through Pseudo-Random Graphs", IEEE Transactions on Information Theory, Vol. 38, 1992, pp. 509-516.

[AGHP] N. Alon, O. Goldreich, J. Hastad and R. Peralta, "Simple constructions of almost k-wise independent random variables", Random Structures and Algorithms, Vol. 3, 1992, pp. 289-304.

[ALMSS] S. Arora, C. Lund, R. Motwani, M. Sudan and M. Szegedy, "Proof verification and hardness of approximation problems", Proceedings of 33rd Annual Symposium on Foundations of Computer Science, 1992, pp. 14-23.

[AMN] Y. Azar, R. Motwani and J. Naor, "Approximating probability distributions using small sample spaces", Unpublished manuscript.

[AS1] N. Alon and J. Spencer, **The probabilistic method**, John Wiley and Sons Inc., 1992.

[AS2] S. Arora and S. Safra, "Probabilistic checking of proofs: a new characterization of NP", Proceedings of 33rd Annual Symposium on Foundations of Computer Science, 1992, pp. 2-13.

[Be] S. J. Berkowitz, "On computing the determinant in small parallel time using a small number of processors", Information Processing Letters, Vol. 18, 1984, pp. 147-150.

[Bec] J. Beck, "An algorithmic Approach to the Lovász local lemma", Random Structures and Algorithms, Vol. 2, 1991, pp. 343-365.

[BR] B. Berger and J. Rompel, "Simulating $(\log^c n)$-wise independence in NC", Journal of the ACM, Vol. 38, 1991, pp. 1026-1046.

[Bi] N. Biggs, **Algebraic graph theory**, Cambridge University Press, 1974.

[Ch] H. Chernoff, "A measure of asymptotic efficiency for tests based on the sum of observations", Annals of Mathematical Statistics, Vol. 23, 1952, pp. 492-509.

[Ch] A.L. Chistov, "Fast parallel calculation on the rank of matrices over a field of arbitrary characteristic", Proc. Intl. Conf. Foundations of Computation Theory, Springer Lecture Notes in Computer Science, Vol. 199, 1985, pp. 63-79.

[Cl] Kenneth L. Clarkson, "Las vegas algorithms for linear and integer programming when the dimension is small", Proceedings of the 4th Annual ACM Symposium on Computational Geometry, 1988, pp. 1-11.

[CGHFRS] B. Chor, O. Goldreich, J. Hastad, J. Friedman, S. Rudich and R. Smolensky, "t-Resilient functions", Proceedings of the 26th Annual Symposium on the Foundations of Computer Science, 1985, pp. 396-407.

[CoWi] D. Coppersmith, S. Winograd, "Matrix multiplication via arithmetic progressions", Proceedings of 28th Annual ACM Sympium on Theory of Computing, 1987, pp. 1-6.

[Cs] L. Csanky, "Fast parallel matrix invertion algorithms", SIAM J. Computing, Vol. 5, 1976, pp.618-623.

[CW] A. Cohen and A. Wigderson, "Dispersers, deterministic amplification and weak random sources", Proceedings of the 30th IEEE Symposium on Foundations of Computer Science, 1989, pp. 14-19.

[Di] P. Diaconis, **Group representations in probability and statistics**, Institute of Mathematical Statistics, Hayward, CA, 1988.

[Er] P. Erdös, "Some remarks on the theory of graphs," Bulletin of the American Mathematics Society, Vol. 53, 1947, pp. 292-294.

[EGLNV] G. Even, O. Goldreich, M. Luby, N. Nissan and B. Veličković, "Approximations of general independent distributions", Proceedings of 24th Annual ACM Symposium on Theory of Computing, 1992, pp. 10-16.

[Fe] W. Feller, **An introduction to the theory of probability and its applications**, John Wiley and Sons, 1967.

[Fr] R. Freivalds, "Fast probabilistic algorithms", Springer Verlag Lecture Notes in CS – Mathematical Foundations of CS, Vol. 74, 1979, pp. 57-69.

[FW] P. Frankl and R.M. Wilson, "Intersecting theorems with geometric consequences," Combinatorica, Vol. 1, pp. 357-368.

[GG] O. Gaber and Z. Galil, "Explicit construction of linear size superconcentrators", Journal of Computer and System Sciences, Vol. 22, 1981, pp. 407-420.

[IZ] R. Impagliazzo and D. Zuckerman, "Recycling random bits", Proceedings of the 30th IEEE Symposium on Foundations of Computer Science, 1989, pp. 248-253.

[Ka] Howard J. Karloff, **Linear programming**, Birkhäuser, 1991.

[KLM] R.M. Karp, M. Luby, N. Madras, "Monte-Carlo approximation for enumeration problems", J. of Algorithms, Vol. 10 , 1989 , pp. 429-448.

[KLNS] J. Kahn, N. Linial, N. Nisan and M. Saks, "On the cover time of random walks in graphs", Journal of Theoretical Probability, Vol. 2, 1989, pp. 121-128.

[KUW1] R. M. Karp, E. Upfal and A. Wigderson, "Constructing a perfect matching is in random NC", *Combinatorica*, Vol. 6, 1986, pp 35-48.

[KUW2] R. M. Karp, E. Upfal and A. Wigderson, "The complexity of parallel search", Journal of Computer and System Sciences, Vol 36, 1988, pp. 225-253.

[KW] R. M. Karp and A. Wigderson, "A fast parallel algorithm for the maximal independent set problem", Journal of the ACM, Vol. 32, 1985, pp. 762-773.

[Li] T. Lindvall, **Lecture notes on the coupling method**, John Wiley & sons, Inc., 1992.

[LPS] A. Lubotzky, R. Phillips and P. Sarnak, "Explicit expanders and the Ramanujan conjecture", Proceedings of the 18th annual ACM Symposium on Theory of Computing, 1986, pp. 240-246.

[Li] N. Linial, "Locality in distributed computing", SIAM Journal on Computing, Vol. 21, 1992, pp. 193-201.

[Lo] L. Lovàsz, "On the ratio of optimal integral and fractional covers", Discrete Math, Vol. 13, 1975, pp. 383-390.

[Lu1] M. Luby, "A simple parallel algorithm for the maximal independent set", SIAM J. on Computing, Vol. 15, 1986, pp. 1036-1053.

[Lu2] M. Luby, “Removing randomness in parallel computation without a processor penalty”, Proceedings of the 29th Annual Symposium on Foundations of Computer Science, 1988, pp. 162-173.

[Ma] G. A. Margulis, “Explicit group-theoretical constructions of combinatorial schemes and their applications to the design of expanders and super concentrators”, Problemy Peredachi Informatsii, Vol. 24, (1988), pp. 51-60. (English translation in Problems of Information Transmission, Vol. 24, pp. 39-46).

[Mu] K. Mulmuley, “A fast parallel algorithm to compute the rank of a matrix over an arbitrary field”, Combinatorica, Vol. 7, 1987, pp. 101-104.

[MNN] R. Motwani, J. Naor and M. Naor, “The probabilistic method yields deterministic parallel algorithms”, Proceedings of the 30th Annual Symposium on the Foundations of Computer Science, 1989, pp. 8-13. To appear, Journal of Computer and System Sciences.

[MR] R. Motwani and P. Raghavan, **Randomized algorithms**, Cambridge University Press, 1994 (in press).

[MS] F. J. MacWilliams and N. J. A. Sloane, **The theory of error correcting codes**, North Holland, Amsterdam, 1977.

[MVV] K. Mulmuley, U.V. Vazirani, V.V. Vazirani, “Matching is as easy as matrix inversion”, Combinatorica, Vol. 7, 1987, pp. 105-114.

[Na] M. Naor, “Constructing Ramsey graphs from small probability spaces”, IBM Research Report RJ 8810, 1992.

[NN] J. Naor and M. Naor, “Small-bias probability spaces: efficient constructions and applications”, SIAM Journal on Computing, Vol. 22, No. 4, 1993, pp. 838-856.

[Pf] P. E. Pfeiffer, **Probability for applications**, Springer Verlag, New York, 1977.

[PY] C.H. Papadimitriou, M. Yannakakis, “The complexity of restricted spanning tree problem”, Journal of the ACM, Vol. 29, 1982, pp. 285-309.

[Ra] P. Raghavan, "Probabilistic construction of deterministic algorithms: approximating packing integer programs", Journal of Computer and System Sciences, Vol. 37, 1988, pp. 130-143.

[RS] J. H. Reif and S. Sen, "Optimal randomized parallel algorithms for computational geometry", Proceedings 16th International Conference on Parallel Processing, St. Charles, IL, 1987. Full version: Duke University Tech. Report, CS-88-01, 1988.

[SB] G. Seroussi and N.H. Bshouty, "Vector sets for exhaustive testing of logic circuits", IEEE Trans. on Information Theory, Vol. 34, 1988, pp. 513-522.

[Sc] J.T. Schwartz, "Fast probabilistic algorithms for verification of polynomial identities", Journal of the ACM, Vol. 27, 1980, pp. 701-717.

[Se] Raimund Seidel, "Small-dimensional linear programming and convex hulls made easy", Discrete Comput. Geom., Vol. 6, 1991, pp. 423-434.

[SS] E. Shamir and J. Spencer, "Sharp concentration of the chromatic number in random graphs $G_{n,p}$", Combinatorica, Vol. 7, (1987), pp. 121-130.

[Sp] J. Spencer, **Ten lectures on the probabilistic method**, SIAM (Philadelphia), 1987.

[Sp1] J. Spencer, "Six standard deviations suffice", Trans. American Mathematical Society, Vol. 289, 1985, pp 679-706.

[Ta] R. M. Tanner, "Explicit construction of concentrators from generalized N-gons", SIAM J. of Alg. Disc. Methods, Vol. 5, 1984 pp. 287-293.

[Va1] L. Valiant , "The complexity of enumeration and reliability problems" , SIAM J. Comput. Vol. 8, 1979 , pp. 410-421.

[Va2] L. Valiant, "Short monotone formulae for the majority function", Journal of Algorithms, Vol. 5, 1984, pp. 363-366.

[Vaz] U. Vazirani, Randomness, adversaries and computation, Ph.D. Thesis, University of California, Berkeley, 1986.

[Zu] D. Zuckerman, “NP-Complete problems have a version that is hard to approximate”, unpublished manuscript, 1992.

[ZG] Y. Zalcstein, M. Garzon, “An NC^2 algorithm for testing similarity of matrices”, Information Processing Letters, Vol. 30, 1989, pp. 253-254.