

Probability in computing

Péter Gács

Computer Science Department
Boston University

Spring 07

Course structure: see the syllabus on
www.cs.bu.edu/~gacs/courses/cs537.

Event space, probability space

Event space: a pair (Ω, \mathcal{A}) where Ω is a set. Elements $\omega \in \Omega$ are called **outcomes**. Further, \mathcal{A} is a set of subsets of Ω called **events**.

Example 1

If Ω is a countable set then we will always assume that $\mathcal{A} = 2^\Omega$, that is every subset of Ω is an event.

In general, we assume that \mathcal{A} is an **algebra**: \emptyset, Ω are in \mathcal{A} and if E, F are in \mathcal{A} then so is $E \setminus F$ (and then of course $E \cup F$ and $E \cap F$).

Example 2

$\Omega = \mathbb{R}$ and \mathcal{A} is the set of all unions of a finite set of intervals (closed, open, half-closed).

Remark: normally it is also required that \mathcal{A} is closed with respect to countable union, but we do not need this in the course.

Given an event space (Ω, \mathcal{A}) and a function $\mathbf{P} : \mathcal{A} \rightarrow [0, 1]$, we call \mathbf{P} a **probability measure** if $\mathbf{P}(\Omega) = 1$ and for disjoint events E, F we have $\mathbf{P}(E \cup F) = \mathbf{P}(E) + \mathbf{P}(F)$. In this case, the triple

$$(\Omega, \mathcal{A}, \mathbf{P})$$

is called a **probability space**.

Examples 3

- Ω is a finite set, $\mathcal{A} = 2^\Omega$ and $\mathbf{P}(E) = |E|/|\Omega|$.
- $\Omega = \mathbb{R}^2$, (plane). Let \mathcal{A} be the smallest algebra containing all the rectangles. Let $f : \mathbb{R}^2 \rightarrow \mathbb{R}$ be an integrable function with $\int_{\mathbb{R}^2} f(x) dx = 1$. For all elements $E \in \mathcal{A}$ we defined $\mathbf{P}(E) = \int_E f(x) dx$. In this case we say that $f(x)$ is the **density function** of the probability measure \mathbf{P} .

Conditional probability

Let $(\Omega, \mathcal{A}, \mathbf{P})$ be a probability space and E, F events. If $\mathbf{P}(E) > 0$ then $\mathbf{P}(F | E) = \mathbf{P}(E \cap F) / \mathbf{P}(E)$ is called the **conditional probability** of event F with respect to E .

If E_1, \dots, E_n are disjoint events with $\bigcup_i E_i = \Omega$ then we will call the set $\{E_1, \dots, E_n\}$ a **partition**. If $\{E_1, \dots, E_n\}$ is a partition and F is an event then the following relations will be often used:

$$\begin{aligned} F &= (F \cap E_1) \cup \dots \cup (F \cap E_n), \\ \mathbf{P}(F) &= \mathbf{P}(F \cap E_1) + \dots + \mathbf{P}(F \cap E_n), \\ &= \mathbf{P}(F | E_1)\mathbf{P}(E_1) + \dots + \mathbf{P}(F | E_n)\mathbf{P}(E_n). \end{aligned}$$

Example 4

Let X be a number that is prime with probability 0.1, pseudo-prime with probability 0.2 and other with probability 0.7. We perform a prime test which is always correct when the number is prime, gives wrong answer with probability 0.02 if the number is pseudoprime and with probability 0.01 if the number is other. What is the probability of wrong answer?

$$\begin{aligned} & \mathbf{P}(\text{wrong}) \\ &= \mathbf{P}(\text{wrong} \mid \text{prime})\mathbf{P}(\text{prime}) \\ &+ \mathbf{P}(\text{wrong} \mid \text{ps-prime})\mathbf{P}(\text{ps-prime}) \\ &+ \mathbf{P}(\text{wrong} \mid \text{other})\mathbf{P}(\text{other}) \\ &= 0.1 \cdot 0 + 0.2 \cdot 0.02 + 0.7 \cdot 0.01. \end{aligned}$$

Independence

Events A_1, \dots, A_n are **independent** when for all every possible subsequence $1 \leq i_1 < \dots < i_k \leq n$ we have

$$\mathbf{P}(A_{i_1} \cap A_{i_2} \cap \dots \cap A_{i_k}) = \mathbf{P}(A_{i_1})\mathbf{P}(A_{i_2}) \cdots \mathbf{P}(A_{i_k}).$$

In particular, A, B are independent if $\mathbf{P}(A \cap B) = \mathbf{P}(A)\mathbf{P}(B)$. If $\mathbf{P}(B) > 0$ this can be written as

$$\mathbf{P}(A \mid B) = \mathbf{P}(A).$$

Note that **pairwise independence is weaker**.

Example 5

Toss an unbiased coin twice, let A say that the first toss is head, B that the second toss is head and C that both are heads. These three events are only pairwise independent.

Random variables

For a probability space $(\Omega, \mathcal{A}, \mathbf{P})$, a **random variable** is a function $X : \Omega \rightarrow \mathbb{R}$ with the property that each set of the form $\{\omega : X(\omega) < a\}$ and $\{\omega : X(\omega) > a\}$ is an event. We will write

$$\mathbf{P}[X < a] = \mathbf{P}\{\omega : X(\omega) < a\}.$$

If X is a random variable then the function

$$F(a) = \mathbf{P}[X < a]$$

is called the **distribution function** of X .

Let X_1, \dots, X_n be random variables on the same probability space. We say that they are **independent** if for all sequences $a_1, \dots, a_n \in \mathbb{R}$ the events

$$\{\omega : X_1(\omega) < a_1\}, \dots, \{\omega : X_n(\omega) < a_n\}$$

are independent.

Example 6

We toss a die twice. The outcomes ω are $(1, 1), (1, 2), \dots, (6, 6)$ with probabilities $1/36$ for each. Let X_1 be value of the outcome of the first toss, and X_2 the value of the second toss. Then X_1, X_2 are independent. On the other hand, X_1 is not independent of $X_1 + X_2$.

Note: if X_1, \dots, X_n are independent and f_i are any functions then $f_1(X_1), \dots, f_n(X_n)$ are also independent. Also, if X, Y, Z are independent and $f(x), g(y, z)$ are functions then $f(X), g(Y, Z)$ are independent.

Expected value

Suppose that X is a random variable that takes on a discrete set of values a_1, a_2, \dots with probabilities p_1, p_2, \dots . Then its expected value is defined as

$$\mathbf{E}X = p_1 a_1 + p_2 a_2 + \dots .$$

Does not always exist.

Example 7

Tossing a die a single time: the expected value is

$$1 \cdot (1/6) + \dots + 6 \cdot (1/6).$$

For a probability space $\Omega = \mathbb{R}^d$ with a density function $p(x)$ and a random variable $X(\omega)$ we define $\mathbf{E}X = \int X(\omega)p(\omega)d\omega$.

Example 8

Consider a fair game of tossing a coin repeatedly and betting on head each time. Play until you win, doubling your bet in every step: in step n it is 2^n .

Expected win is 1.

Expected maximum loss before winning is ∞ .

The expected value is **additive**:

$$\mathbf{E}(\alpha X + \beta Y) = \alpha \mathbf{E}X + \beta \mathbf{E}Y,$$

even if X, Y are not independent.

Example 9

We toss a die twice, and win the sum of the points if the two tosses are equal. Let $X_i = 2i$ if both tosses are equal to i . Then $\mathbf{E}X_i = (1/36)(2i)$. Our expected win is

$$\mathbf{E}(X_1 + \cdots + X_6) = \mathbf{E}X_1 + \cdots + \mathbf{E}X_6 = (1/36) \cdot 2 \cdot (1 + \cdots + 6).$$

If X_1, \dots, X_n are independent random variables then the expected value is also **multiplicative**:

$$\mathbf{E}(X_1 \cdots X_n) = (\mathbf{E}X_1) \cdots (\mathbf{E}X_n).$$

Converse: if

$$\mathbf{E}(f_1(X_1) \cdots f_n(X_n)) = (\mathbf{E}f_1(X_1)) \cdots (\mathbf{E}f_n(X_n))$$

for all functions f_1, \dots, f_n then X_1, \dots, X_n is independent.

Markov inequality

An important tool for estimating probabilities with the help of expectations. Let X be a nonnegative random variable, $\lambda > 0$, then

$$\mathbf{P}[X > \lambda \mathbf{E}] < \frac{1}{\lambda}.$$

Inner product

Random variables over a probability space can be viewed as vectors: there is an operation of linear combination $\lambda X + \mu Y$.

Subspace: those random variables X with $\mathbf{E}X = 0$.

We introduce an **inner product**:

$$\langle X, Y \rangle = \mathbf{E}XY.$$

This behaves just like the inner product of vectors: it is bilinear, and $\langle X, X \rangle \geq 0$ with $\langle X, X \rangle = 0$ iff $\mathbf{P}[X = 0] = 1$.

For random variables X_1, X_2 with $\mathbf{E}X_i = \mu_i$ we define the **covariance**

$$\mathbf{Cov}(X_1, X_2) = \langle X_1 - \mu_1, X_2 - \mu_2 \rangle = \mathbf{E}(X_1 - \mu_1)(X_2 - \mu_2).$$

Two random variables are **orthogonal** if $\langle X, Y \rangle = 0$. Typically, we will apply this notion only in the subspace of variables with expected value 0.

Two variables are **uncorrelated** if their covariance is 0.

If X, Y are independent then they are uncorrelated. Indeed, let $X' = X - \mathbf{E}X, Y' = Y - \mathbf{E}Y$. Then $\mathbf{E}X' = \mathbf{E}Y' = 0$, and X', Y' are independent. therefore $\mathbf{E}(X'Y') = (\mathbf{E}X')(\mathbf{E}Y') = 0$.

The converse does not always hold. But if $f(X), g(Y)$ are uncorrelated for all f, g then X, Y are independent.

The converse also holds if X, Y only take 2 values: Let $X = 1$ with probability p and 0 with probability $(1 - p)$. Let $Y = 1$ with probability q and 0 with probability $(1 - q)$. Then $X' = X - p, Y' = Y - q$. Then the converse also holds: if X', Y' are orthogonal then X', Y' (and thus X, Y) are independent. Indeed,

$$\mathbf{E}(X - p)(Y - q) = \mathbf{E}XY - p\mathbf{E}Y - q\mathbf{E}X + pq = \mathbf{E}XY - pq.$$

If this is 0 then $pq = \mathbf{E}XY = \mathbf{P}[X = 1 \wedge Y = 1]$, showing that X, Y are independent.

Theorem 10 (Cauchy-Schwartz)

$$\langle X, Y \rangle^2 \leq \langle X, X \rangle \langle Y, Y \rangle.$$

Thus

$$(\mathbf{E}XY)^2 \leq \mathbf{E}X^2 \mathbf{E}Y^2.$$

If X is a random variable $\mu = \mathbf{E}X$, with $X' = X - \mu$ then we define

$$\mathbf{Var}X = \mathbf{Cov}(X, X) = \mathbf{E}(X - \mu)^2.$$

Equivalent expression:

$$\mathbf{Var}X = \mathbf{E}X^2 - \mu^2.$$

Since $\mathbf{Var}(\lambda X) = \lambda^2 \mathbf{Var}(X)$, we call $\sqrt{\mathbf{Var}X}$ the **standard deviation**.

Theorem 11

$$\mathbf{Var}(X + Y) = \mathbf{Var}(X) + \mathbf{Var}(Y) + 2\mathbf{Cov}(X, Y).$$

Thus if $\mathbf{Cov}(X, Y) = 0$ then $\mathbf{Var}(X + Y) = \mathbf{Var}(X) + \mathbf{Var}(Y)$. In particular, this is true if X, Y are independent.

Application: let X_1, \dots, X_n be orthogonal, identically distributed, with $\sigma^2 = \mathbf{Var}X_j$. Then

$$\mathbf{Var}(X_1 + \dots + X_n) = n\sigma^2.$$

So the standard deviation will grow only by a factor \sqrt{n} .

Chebyshev inequality

Let $\mu = \mathbf{E}X$, $\sigma^2 = \mathbf{Var}X$. The new inequality comes from the Markov inequality applied to the variable $(X - \mu)^2$. It says, for $\lambda > 0$:

$$\mathbf{P}[|X - \mu| > \lambda\sigma] < \frac{1}{\lambda^2}.$$

Law of large numbers

Let $S_n = X_1 + \cdots + X_n$.

Theorem 12 (Law of large numbers)

For each n , let X_1, \dots, X_n be orthogonal, with $\mathbf{E}X_i = \mu$, $\mathbf{Var}X_i = \sigma^2$. Then for all $\varepsilon > 0$ we have

$$\lim_{n \rightarrow \infty} \mathbf{P}[|S_n/n - \mu| > \varepsilon] = 0.$$

Indeed, by Chebyshev's inequality for $\lambda > 0$:

$$\begin{aligned}\mathbf{P}[|S_n - n\mu| > \lambda\sigma\sqrt{n}] &< 1/\lambda^2, \\ \mathbf{P}[|S_n/n - \mu| > \lambda\sigma/\sqrt{n}] &< 1/\lambda^2, \\ \mathbf{P}[|S_n/n - \mu| > \varepsilon] &< \frac{\sigma^2}{n\varepsilon^2}\end{aligned}$$

where we chose $\lambda = \varepsilon\sqrt{n}/\sigma$.

Thus, for large λ the value S_n is mostly in the interval $n\mu \pm \lambda\sigma\sqrt{n}$.

Example 13

Let $X_i = 1$ with probability p and 0 otherwise: when a biased coin falls head above. Then $\mathbf{E}X_i = p$, $\mathbf{Var}X_i = p(1 - p)$. Further S_n is the number of heads in n tosses. The inequality says that with large probability this number is within

$$np \pm \lambda\sqrt{np(1 - p)}.$$

This interprets probability in terms of **relative frequency**.

Can we make the interval tighter than $\pm c\sqrt{n}$ in the law of large numbers? No, this is shown by the **central limit theorem** (which we will not use) that tells us that (under conditions that for us are always satisfied) $\frac{S_n - \mu n}{\sigma\sqrt{n}}$ for independent X_1, \dots, X_n has a limit distribution (not concentrated on 0). It implies that, for example $\mathbf{P}[S_n - \mu n > \sigma\sqrt{n}]$ converges to a constant > 0 .

Exponential convergence

Can we make the $O(1/n)$ convergence in the law of large numbers faster? Yes, there are theorems of the sort

$$\mathbf{P}[|S_n/n - \mu| > \varepsilon] < e^{-nf(\varepsilon)}.$$

Here, we just want S_n/n to be in a constant-size interval around μ , but this should fail only with exponentially small probability. One such theorem is the so-called “Chernoff bound”. Assume $0 \leq X_i \leq 1$, then

$$\mathbf{P}[S_n/n - \mu > \varepsilon] < e^{-2\varepsilon^2 n},$$

$$\mathbf{P}[\mu - S_n/n > \varepsilon] < e^{-2\varepsilon^2 n},$$

Let us see how to get such bounds.

Assume X_i independent with $0 \leq X_i \leq 1$ (if this is not the case, rescale), $\mu_i = \mathbf{E}X_i$, $\mu = \frac{1}{n} \sum_i \mu_i$.

Theorem 14 (Law of large numbers with exponential convergence)

Let $\lambda - \mu = \varepsilon$, then we have

$$\lambda > \mu \Rightarrow \mathbf{P}[S_n/n > \lambda] \leq 2^{nH_\mu(\lambda)} \leq e^{-2n\varepsilon^2},$$

$$\lambda < \mu \Rightarrow \mathbf{P}[S_n/n < \lambda] \leq 2^{nH_\mu(\lambda)} \leq e^{-2n\varepsilon^2},$$

where $H_\mu(\lambda) = \lambda \log \frac{\lambda}{\mu} + (1 - \lambda) \log \frac{1 - \lambda}{1 - \mu} < 0$ if $\lambda \neq \mu$.

For the proof, fix a positive number b , and compute, using **independence**:

$$\mathbf{E}b^{S_n} = \mathbf{E}b^{X_1} \dots \mathbf{E}b^{X_n} = (\mathbf{E}b^{X_1})^n.$$

For $\mu < \lambda < 1$, choose $b > 1$, then $S_n > \lambda n \Leftrightarrow b^{S_n} > b^{\lambda n}$. For $0 < \lambda < \mu$, choose $b < 1$, then $S_n < \lambda n \Leftrightarrow b^{S_n} > b^{\lambda n}$. In both cases, by Markov's inequality,

$$\mathbf{P}[S_n > \lambda n] = \mathbf{P}[b^{S_n} > b^{\lambda n}] < b^{-\lambda n} \mathbf{E}b^{S_n} = b^{-\lambda n} \prod_{i=1}^n \mathbf{E}b^{X_i}.$$

To simplify, since b^x is a convex function of x , it is below the chord in $0 \leq x \leq 1$:

$$b^x \leq 1 + x(b - 1).$$

Taking expectation: $\mathbf{E}b^{X_i} \leq 1 + \mu_i(b - 1)$.

By the inequality of arithmetic and geometric mean:

$$\begin{aligned}\prod_i (1 + \mu_i(b - 1)) &\leq \left(\frac{1}{n} \sum_i (1 + \mu_i(b - 1)) \right)^n \\ &= (1 + \mu(b - 1))^n.\end{aligned}$$

Multiplying with $b^{-\lambda}$, our bound is $(g(b))^n$ where

$$g(b) = b^{-\lambda}(1 + \mu(b - 1)) = (1 - \mu)b^{-\lambda} + \mu b^{1-\lambda},$$

a convex function. Keeping μ fixed, choose b to minimize this by differentiation:

$$b^* = \frac{\lambda(1 - \mu)}{(1 - \lambda)\mu'}$$

which is indeed < 1 if $\lambda < \mu$ and > 1 if $\lambda > \mu$.

Substitution gives

$$g(b^*) = \left(\frac{\mu}{\lambda}\right)^\lambda \left(\frac{1-\mu}{1-\lambda}\right)^{1-\lambda} \leq \lambda \left(\frac{\mu}{\lambda}\right) + (1-\lambda) \left(\frac{1-\mu}{1-\lambda}\right) = 1.$$

The (arithmetic-geometric) inequality is strict unless $\lambda = \mu$. We can write $g(b^*) = 2^{H_\mu(\lambda)}$, where

$$H_\mu(\lambda) = \lambda \log \frac{\mu}{\lambda} + (1-\lambda) \log \frac{1-\mu}{1-\lambda}.$$

The two “Chernoff bounds” mentioned above are derived from this in a paper of Hoeffding. Just as others in your books, they come from analyzing this (concave) function of λ around $\lambda = \mu$ or choosing a more convenient (even if not optimal) value for b .

Another useful form For $\lambda > \mu$ it can be shown that

$$H_{\mu}^{(e)}(\lambda) = \lambda \ln \frac{\mu}{\lambda} + (1 - \lambda) \ln \frac{1 - \mu}{1 - \lambda} < \lambda \ln \frac{\mu}{\lambda} + \lambda.$$

Indeed, we can ignore the $1 - \mu$ and use the inequality $(1 - \lambda) \ln \frac{1}{1 - \lambda} \leq \lambda$ (homework). This results in the upper bound

$$\mathbf{P}[S_n/n > \lambda] < \left(\frac{e\mu}{\lambda}\right)^{\lambda n},$$

which is useful only if $\lambda > e\mu$.

Binomial coefficients

Let $X_i = 1$ with probability $1/2$ and 0 otherwise, then $\mu = 1/2$. Let $\lambda < 1/2$. Then

$$2^{-n} \sum_{k \leq \lambda n} \binom{n}{k} = \mathbf{P}[S_n < \lambda n] \leq 2^{nH_\mu(\lambda)}.$$

Now,

$$H_\mu(\lambda) = \lambda \log \frac{1/2}{\lambda} + (1 - \lambda) \log \frac{1/2}{1 - \lambda} = H(\lambda) - 1,$$

where $H(\lambda) = -\lambda \log \lambda - (1 - \lambda) \log(1 - \lambda)$ is the ordinary entropy function.

It follows that

$$\sum_{k \leq \lambda n} \binom{n}{k} \leq 2^{nH(\lambda)},$$

a very useful inequality. For small λ , we can again use the inequality

$$(1 - \lambda) \ln \frac{1}{1 - \lambda} \leq \lambda,$$

which gives with $m = \lambda n$:

$$\sum_{k \leq \lambda n} \binom{n}{k} \leq \left(\frac{e}{\lambda}\right)^{n\lambda}, \quad \sum_{k \leq m} \binom{n}{k} \leq \left(\frac{ne}{m}\right)^m.$$

Transformation to other range

Suppose we want $-1 \leq X_i \leq 1$. Then we can apply the bound to $X'_i = (X_i + 1)/2$, that is $X_i = 2X'_i - 1$. If $S_n/n > \mu + \varepsilon$ then $2S'_n - 1 > 2\mu' - 1 + \varepsilon$, that is $S'_n > \mu' + \varepsilon/2$. By the Chernoff bound this has probability bound

$$e^{-2(\varepsilon/2)^2 n} = e^{-(1/2)\varepsilon^2 n}.$$

Application to the set balancing problem

We have a set U , and subsets $A_1, \dots, A_n \subseteq U$. We are looking for a set B that the (maximum) discrepancy

$$\max_{i=1}^n \left| |A_i \cap B| - |A_i \setminus B| \right|$$

is minimal.

Matrix language: given an $n \times m$ matrix A with 0-1 entries. We are looking for a vector \mathbf{b} with 1, -1 entries for which

$$\|A\mathbf{b}\|_\infty$$

is minimal.

Let us see how good a vector we can get by random choice. Let β_1, \dots, β_n be independent, with $\mathbf{P}[\beta_j = 1] = \mathbf{P}[\beta_j = -1] = 1/2$, and let $\boldsymbol{\beta} = (\beta_1, \dots, \beta_n)$. Let \mathbf{a}_i^T the i th row, and let j_1, \dots, j_k be the positions of the 1's in it. Then

$$\mathbf{a}_i^T \boldsymbol{\beta} = \beta_{j_1} + \dots + \beta_{j_k},$$

a sum of k independent random variables. Let us fix a parameter t , to be chosen later. Two cases.

- $k \leq t$. Then $|\mathbf{a}_i^T \boldsymbol{\beta}| \leq t$.
- $k > t$. By the Chernoff bound, since here $\mu = 0$,

$$\mathbf{P}[|\mathbf{a}_i^T \boldsymbol{\beta}| > t] \leq 2e^{-\frac{1}{2}(t/k)^2k} = 2e^{-\frac{1}{2}t^2/k} \leq 2e^{-t^2/(2m)}.$$

Choose $t = \sqrt{4m \ln n}$, then this is $2/n^2$. So with probability $1 - 2/n$, all n rows have discrepancy $\leq t$.

Can one do better?

Yes, but not by just random choice. Spencer achieves a discrepancy of

$$O\left(\sqrt{m \ln(n/m)}\right).$$

(See the Alon-Spencer book.) For $m = n$ this is $O(\sqrt{n})$ while the random choice gives $O(\sqrt{n \ln n})$

Randomization or average case

Algorithms can be analyzed probabilistically from several points of view. First distinction:

- 1 The algorithm is deterministic, but we analyze it on random inputs.
- 2 We introduce randomness during computation, but the input is fixed.
- 3 Randomize and also analyze on random inputs.

Approach 1 is less frequently used, since we rarely have reliable information about the distribution of our inputs. Levin's theory of problems that are hard on average addresses general questions of this type.

Most practical uses of randomness belong to category 2, randomization.

Example 15 (Quicksort)

The deterministic quicksort algorithm has a quadratic worst-case performance. Its average-case performance is good, but there is no reason to assume that the permutation we have to sort is “random”. In practice, getting a reversely ordered file is more likely than getting a completely disordered one.

On the other hand, a randomized version provides us the same benefits as a random input would (see below).

Example 16 (Simplex algorithm)

It has been known for some time, that the simplex algorithm of linear programming performs well on random inputs. But it is much less clear, how to turn this observation into a well-performing randomized version of the simplex method.

Given an array $A = (a_1, \dots, a_n)$, we want to sort it using comparisons.
The **recursive** algorithm

Quicksort(i, k, A).

sorts elements a_i, \dots, a_j **if** $i \leq k$. It uses a subroutine called

Partition(i, j, k, A), $i < k, i \leq j \leq k$.

This will take **pivot element** a_j and rearrange a_i, \dots, a_k comparing them with a_j , putting all elements less than a_j before it and all elements greater after it.

Now Quicksort(i, k, A) works as follows. It does nothing if $i \geq k$, else it chooses some element $i \leq j \leq k$ and returns $(A', j') = \text{Partition}(i, j, k, A)$ where j' is the new position of a_j and A' is the new order. Then it applies Quicksort($i, j' - 1, A'$) and Quicksort($j' + 1, k, A'$).

Analysis (following Levin)

In Quicksort($1, n, A$), eventually every element will be chosen as a pivot. Let A^* be the final array. Assume that pivots are chosen randomly. For $i < j$, let $X_{ij} = 1$ if a_i^* will be compared to a_j^* , and 0 otherwise. The expected number of comparisons is $\sum_{i < j} \mathbf{E}X_{ij}$. Let $\tau(i)$ denote the time when a_i^* was chosen and L the length of the interval to be partitioned at this time. For $i < j$, let $\tau^*(i, j) = \min(\tau(i), \tau(i+1), \dots, \tau(j))$. Then $X_{ij} = 1$ iff $\tau^*(i, j) = \min(\tau(i), \tau(j))$. For each $t = 1, \dots, n$ with $\mathbf{P}[X_{ij} = 1] > 0$ we have $\mathbf{P}[X_{ij} = 1 \mid \tau^*(i, j) = t] \leq \frac{2}{j-i+1}$, hence

$$\begin{aligned}
\mathbf{E}X_{ij} &= \mathbf{P}[X_{ij} = 1] \\
&= \sum_{t>0, l \geq j-i+1} \mathbf{P}[\tau^*(i, j) = t, L = l] \mathbf{P}[X_{ij} = 1 \mid \tau^*(i, j) = t, L = l] \\
&= \sum_{t>0, l \geq j-i+1} \mathbf{P}[\tau^*(i, j) = t, L = l] \frac{2}{l} \leq \frac{2}{j-i+1}.
\end{aligned}$$

Hence

$$\begin{aligned}
\frac{1}{2} \sum_{i < j} \mathbf{E}X_{ij} &\leq \sum_{i=1}^{n-1} \sum_{j=i+1}^n \frac{1}{j-i+1} = \sum_{i=1}^{n-1} \sum_{k=2}^{n-i+1} \frac{1}{k} = \sum_{k=2}^n \sum_{i=1}^{n-k+1} \frac{1}{k} \\
&= \sum_{k=2}^n \frac{n+1-k}{k} \leq (n-1) \sum_{i=2}^n \frac{1}{k} = n \ln n + O(n).
\end{aligned}$$

Is this all we want to know?

No, an estimate on the expected value may still allow very large fluctuations. By Markov's inequality we know only that if N is the number of comparisons then, say, $\mathbf{P}[N > 20n \ln n] < 1/20$. To know more about the **concentration** of N around its expected value, we may need to know more about, say, $\mathbf{Var}N$. We do not have exponential convergence like in Chernoff bound, but it is better than polynomial (McDiarmid-Hayward 96).

What questions to ask about a randomized algorithm? Clearly, with randomization we give up (almost always) some certainty, but what sort?

- 1 If an algorithm always solves the problem exactly (like Quicksort), it be called a **Las Vegas** algorithm (for no particular reason) as opposed to Monte-Carlo algorithm for the case when the result may be wrong. In the Las Vegas case, we want to know, how fast (in various statistical measures)?
- 2 Otherwise we will generally assume a fixed time bound and are interested in the probability of correct solution, or other statistical goodness of the solution.

Let Σ be a finite alphabet. We will consider **languages**, sets $L \subseteq \Sigma^*$ of strings with letters in Σ .

Definition 17

For a language $L \subseteq \Sigma^*$ we say that $L \in ZP(t(n))$ if there is a Las Vegas algorithm $A(x)$ working in time $t(n)$ deciding $x \in L$ in expected time $O(t(n))$. Let $ZPP = \bigcup_k ZP(n^k)$.

Example?

It is not easy to show a nontrivial example of a ZPP language. Adleman and Huang have shown that prime testing is in ZPP, but now it is also known that it is in P.

One-sided error

Matrix product testing

Given $n \times n$ integer matrices A, B , there is no known algorithm to compute AB in $O(n^2)$ algebraic operations. Even if we are given a matrix C , there is no known deterministic algorithm to **test** the equality $AB = C$.

The following algorithm will accept equality and reject inequality with probability $\geq 1/2$. Repeating it k times will reduce the probability of false positive to 2^{-k} .

Simple randomized algorithm

Choose a random vector x with entries from $\{-1, 1\}$.

Compute $c = Cx$, $b = Bx$, $c' = Ab$. If $c = c'$, accept, else reject.

This algorithm takes $O(n^2)$ operations.

If $C = AB$ then it always accepts. Else, it accepts with probability $\leq 1/2$. Indeed, let $D = C - AB$, and assume $d_{ij} \neq 0$. We have

$$(Dx)_i = \sum_{k \neq j} d_{ik}x_k + d_{ij}x_j.$$

Let x_k for $k \neq j$ be chosen arbitrarily, this fixes the first term of the right-hand side. Now x_j is still chosen independently and uniformly over $\{-1, 1\}$, and one of the two choices makes the result $\neq 0$.

Definition 18

A language L is in $R(t(n))$ if there is a randomized algorithm A working in time $O(t(n))$ such that for all $x \in \Sigma^*$

- if $x \notin L$ then $A(x)$ rejects.
- if $x \in L$ then $A(x)$ accepts with probability $\geq 1/2$.

Let $RP = \bigcup_k R(n^k)$.

If we want $1 - 2^{-k}$ in place of $1/2$, we can repeat k times; this does not change the definition of RP.

Example?

We have not seen an interesting example of an RP language yet, since matrix product testing is also in P.

Such an example will be prime testing, though now it is also known that primes are in P.

Contrast RP with NP

$L \in \text{RP}$ if there is a k and a (deterministic) algorithm $A(x, r)$ running in time n^k with $x \in \Sigma^n, r \in \{0, 1\}^{n^k}$ such that

- if $x \notin L$ then $A(x, r)$ rejects for all r .
- if $x \in L$ then $A(x, r)$ accepts for at least half of all values of r .

On the other hand $L \in \text{NP}$ if there is a k and a (deterministic) algorithm $A(x, r)$ running in time n^k with $x \in \Sigma^n, r \in \{0, 1\}^{n^k}$ such that

- if $x \notin L$ then $A(x, r)$ rejects for all r .
- if $x \in L$ then $A(x, r)$ accepts for at least one value of r .

The algorithm $A(x, r)$ in the NP definition is called the **verifier** algorithm, the values of r for which it accepts are called **witnesses**, or **certificates**. Thus, $\text{RP} \subseteq \text{NP}$.

An NP language L is also in RP if it **has** a verifier algorithm with the property that if x has a witness then it has many (at least half of all potential ones).

Example 19

The word **has** is important in the above remark. The language COMPOSITE is in NP, since a verifier predicate is

$$B(x, r) \Leftrightarrow r|x \wedge r \notin \{1, x\}.$$

For this verifier, however, it is not true that if there is one witness there are many. For example, if $x = p^2$ for a prime p then p is the only witness.

Nevertheless, $\text{COMPOSITE} \in \text{RP}$. But this is shown with the help of a randomized prime/compositeness test algorithm (see later), that is with a **different** verifier $B'(x, r)$ for the same language COMPOSITE.

Two-sided error

It is actually more natural to consider a randomized complexity class with two-sided error.

Definition 20

A language L is in $\text{BP}(t(n))$ if there is a randomized polynomial-time algorithm A working within time $O(t(n))$ such that for all $x \in \Sigma^*$

- if $x \in L$ then $A(x)$ rejects with probability $\leq 1/3$.
- if $x \notin L$ then $A(x)$ accepts with probability $\leq 1/3$.

Let $\text{BPP} = \bigcup_k \text{BP}(n^k)$.

Example?

I do not recall a simple natural example of BPP.

But since RP is not closed under complementation, if $L_1, L_2 \in \text{RP}$ then about $L_1 \setminus L_2$ we can only say that it is in BPP.

Theorem 21

The definition of BPP does not change if we replace $2/3$ with $1/2 - \varepsilon$ for a fixed $\varepsilon > 0$, and also not if we replace it with 2^{-n^k} for any $k > 0$.

To get from error probability $1/2 - \varepsilon$ to error probability 2^{-n^k} , use repetition $\text{const} \cdot n^k$ times, majority voting and the Chernoff bound.

Why not $1/2$?

The definition of BPP does not work with $1/2$ in place of $2/3$: in that case we get a (**probably**) much larger class closely related to $\#P$ (see later). But we could use any $1/2 + \varepsilon$ for some constant ε .

Characterizing Las Vegas

The proof of the following theorem is a good opportunity to practice our notions.

Theorem 22

The following properties are equivalent for a language L :

- (i) $L \in \text{ZPP}$;
- (ii) $L \in \text{RP} \cap \text{co-RP}$;
- (iii) *There is a randomized polynomial-time algorithm that accepts or rejects, correctly, or returns the answer “I give up”, with probability $\leq 1/2$.*

Proof. It is obvious that (i) implies (ii) (Markov inequality). To see that (ii) implies (iii), submit x to a randomized algorithm that accepts L in polynomial time and also to one that accepts $\Sigma^* \setminus L$ in polynomial time. If they give opposite answers then the answer of the first machine is correct. If they give identical answers then we “give up”. In this case, one of them made an error and this has probability at most $1/2$.

To see that (iii) implies (i) modify the algorithm A given in (iii) in such a way that instead of the answer “I give up”, it restarts. If on input x , the number of steps of $A(x)$ is τ and the probability of giving it up is p then on this same input, the expected number of steps of the modified machine is

$$\sum_{t=1}^{\infty} p^{t-1} (1-p) t \tau = \frac{\tau}{1-p} \leq 2\tau.$$



Corollary 23

If a problem has a Las Vegas algorithm $A(x)$ with running time $T(x)$ with $\mathbf{E}T(x) \leq p(n)$ for a polynomial $p(n)$, ($n = |x|$), then it has another one, $A'(x)$, with a constant $\rho < 1$ and random running time $U(x)$, with the property

$$\mathbf{P}\left[\frac{U(x)}{p(n)} > t\right] < \rho^t.$$

In other words, $U(x)$ has very small probability of being large; much better than what comes from the Markov inequality for T . (Then, of course, not just $\mathbf{E}U$ is polynomial but also for example $\mathbf{E}U^2$.)

Relation to P/poly

A computational model similar to P is the set of languages recognizable by a polynomial-size Boolean circuit. It can also be characterized in terms of Turing machines.

Definition 24

A language L is in $\text{Time}(t(n))/f(n)$ if there is Turing machine M such that on inputs x, a of size $n, f(n)$, $M(x, a)$ halts in time $O(t(n))$, and for each n there is a fixed string a_n such that $x \in L$ iff $M(x, a_n)$ accepts. We define $\text{P/poly} = \bigcup_{k,l} \text{Time}(n^k)/n^l = \bigcup_k \text{Time}(n^k)/n^k$.

The string a_n is the **advice** string for n ; it is the same for all inputs x of length n .

Theorem 25

A language is in P/poly if and only if it can be computed by a logic circuit of polynomial size.

The idea of the proof is that for each n the advice string a_n describes the logic circuit that decides $x \in L$ for each n . (The strict proof needs to elaborate the idea in both directions.)

The more interesting theorem is:

Theorem 26 (Adleman)

$BPP \subseteq P/poly$.

Proof. Let $L \in \text{BPP}$, $|x| \in \Sigma^n$, with $|\Sigma| > 1$. A simple idea: the good coin tosses deciding $x \in L$ could play the role of the advice string a_n . But this is too simple, since the good coin tosses may be different for each x . Let M be a probabilistic Turing machine working in time n^k that decides L with error probability $< |\Sigma|^{-n}$. Write $M(x)$ as $M'(x, r)$, where M' is a deterministic Turing machine with input $x \in \Sigma^n$ and auxiliary input $r \in \{0, 1\}^{n^k}$ which represents the coin tosses. Let

$$E(x) = \{ r : M(x, r) \text{ decides incorrectly whether } x \in L \},$$

then $\mathbf{P}(E(x)) < |\Sigma|^{-n}$. Let $E_n = \bigcup_{x \in \Sigma^n} E(x)$, then $\mathbf{P}(E_n) < 1$, therefore there is a string $a_n \in \{0, 1\}^{n^k} \setminus E_n$. Then $M(x, a_n)$ decides $x \in L$ always correctly, so a_n can be taken as the advice string. \square

Adleman's theorem shows that the P/poly is stronger than P and randomization. Is maybe P/poly and randomization even more powerful?

The answer is, no. There is a natural definition of **randomized logic circuits**, namely circuits with some extra inputs for random bits. The reasoning of Adleman's theorem generalizes to this case, showing anything that can be computed by a polynomial-size randomized circuit can be also computed by a (somewhat larger, but still polynomial-size) circuit without randomization.

A **zero-sum two-person game** is played between players R and C (stands for “row and column” and defined by an $m \times n$ matrix A . We say that if player R chooses a **pure strategy** $i \in \{1, \dots, m\}$ and player C chooses pure strategy $j \in \{1, \dots, n\}$ then there is **payoff**: player C pays amount a_{ij} to player R .

Example 27

$m = n = 2$, pure strategies $\{1, 2\}$ are called “attack left”, “attack right” for player R and “defend left”, “defend right” for player C . The matrix is

$$A = \begin{pmatrix} -1 & 1 \\ 1 & -1 \end{pmatrix}.$$

Mixed strategy: a probability distribution over pure strategies.
 $\mathbf{p} = (p_1, \dots, p_m)$ for player R and $\mathbf{q} = (q_1, \dots, q_m)$ for player C.

Expected payoff:

$$\sum_{ij} a_{ij} p_i q_j = \mathbf{p}^T \mathbf{A} \mathbf{q}.$$

If player R knows the mixed strategy \mathbf{q} of player C, he will want to achieve

$$\max_{\mathbf{p}} \sum_i p_i \sum_j a_{ij} q_j = \max_i \sum_j a_{ij} q_j$$

since a pure strategy always achieves the maximum.

Player C wants to minimize this and can indeed achieve

$$\min_q \max_i \sum_j a_{ij} q_j.$$

This can be rewritten as a linear programming problem:

$$\begin{array}{ll} \text{minimize} & t \\ \text{subject to} & t \geq \sum_j a_{ij} q_j, \quad i = 1, \dots, m \\ & q_j \geq 0, \quad j = 1, \dots, n \\ & \sum_j q_j = 1. \end{array}$$

It is straightforward to check that (as for any real function of p, q):

$$\max_p \min_q p^T A q \leq \min_q \max_p p^T A q.$$

But in our case, we will have equality!

The following theorem is a straightforward consequence of linear programming duality.

Theorem 28 (von Neumann)

We have

$$\begin{aligned} \min_q \max_i \sum_j a_{ij} q_j &= \min_q \max_p \sum_j a_{ij} p_i q_j \\ &= \max_p \min_q \sum_j a_{ij} p_i q_j = \max_p \min_j \sum_i a_{ij} p_i. \end{aligned}$$

Geometrically, this says that as a function of the mixed strategies p, q , the function $p^T A q$ has a **global saddle point**, which is the maximum (in p) of minima (in q) as well as the minimum of maxima (where minimization and maximization is both times by the same variables).

Example 29

For the matrix $\mathbf{A} = \begin{pmatrix} -1 & 1 \\ 1 & -1 \end{pmatrix}$ we have, writing $\mathbf{p}^T = (p, 1 - p)$:

$$\begin{aligned} \sum_{ij} a_{ij} p_i q_j &= -pq + p(1 - q) + (1 - p)q - (1 - p)(1 - q) \\ &= -4(p - 1/2)(q - 1/2), \end{aligned}$$

the saddle point is $p = q = 1/2$. (Transforming to $x = p - 1/2$, $y = q - 1/2$, you get $-4xy$, and the function xy is known to have a saddle point at $x = y = 0$.)

Application to algorithms

Yao's theorem will connect the average time of some deterministic algorithms (on varying inputs) with that of a randomized algorithms (on fixed input). Consider a finite number m of possible inputs x_i . and a finite number n of possible computations (algorithms) C_j . Assume that there is some maximum common maximum time T in which each algorithm terminates. Let the row player R choose an input x_i , and the column player C choose an algorithm C_j . The payoff is some aspect

$$a_{ij} = L(C_j(x_i))$$

of the computation $C_j(x_i)$. For definiteness, let it be the **computation time**.

A distribution p on inputs is a **random input**. A distribution q on algorithms a **randomized algorithm**.

Let us apply the minimax theorem:

Theorem 30 (Yao)

We have

$$\max_p \min_j \sum_i L(C_j(x_i)) p_i = \min_q \max_i \sum_j L(C_j(x_i)) q_j.$$

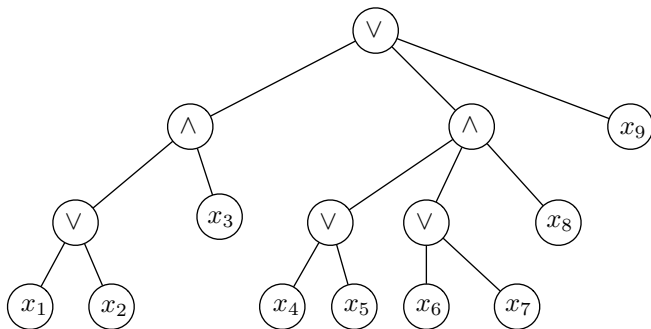
Thus the time we get by choosing an input distribution that maximizes the smallest possible time of all (deterministic) algorithms (in our collection), is the same as the by choosing the randomized algorithm that minimizes the worst expected time on all possible (deterministic) inputs.

In practice, frequently only the easy, weak form of the theorem is used, namely that for each p, q we have

$$\min_j \sum_i L(C_j(x_i)) p_i \leq \max_i \sum_j L(C_j(x_i)) q_j.$$

Game trees or tree formulas

Consider a Boolean formula (formed from \wedge, \vee, \neg) with the properties that **each variable appears at most once**. Using Morgan's laws, push the negations to the lowest level, and if needed rename the variables. What remains is an **and-or** formula, which can be represented as a tree:



- See how the tree divides into even and odd levels. Given the values of the inputs, evaluating the expression can also be seen as finding the value of a two-person game of strategy, (played from top down).
- On the other hand, we can simplify the form of the computation problem using only NOR gates (and again renaming variables). Let us do this, and call our function

$$f(x_1, \dots, x_n).$$

Let us be interested in the **query complexity**: How many **inputs** need to be evaluated to compute the result? For simplicity, assume that the formula is a **full binary tree**, so $n = 2^k$ for a height k .

It is easy to see that each correct algorithm will have to evaluate **all** variables in the **worst case**. But a randomized algorithm may do better. Look at

$$f(x_1, x_2) = x_1 \text{ NOR } x_2.$$

We have $f(x_1, x_2) = 0$ if $x_1 = 1$ or $x_2 = 1$. Evaluate first x_r where r is random in $\{1, 2\}$. Let $r' = 3 - r$.

- If $f(x_1, x_2) = 1$ (though we do not know this yet), then $x_r = 0$ but we need to evaluate $x_{r'}$ also, so the expected number of steps is 2.
- If $f(x_1, x_2) = 0$ then with probability $\geq 1/2$ we get $x_r = 1$ and we can stop. With probability $\leq 1/2$ we need to see $x_{r'}$, too. Hence the expected value is

$$\frac{1}{2} \cdot 2 + \frac{1}{2} \cdot 1 = \frac{3}{2}.$$

So, randomization does not help when $f(x_1, x_2) = 1$.

Let us not give up, look at

$$f(x_1, x_2, x_3, x_4) = y_1 \text{ NOR } y_2,$$

where $y_1 = x_1 \text{ NOR } x_2$, $y_2 = x_3 \text{ NOR } x_4$. In the randomized strategy, first let us compute y_r (using the previous randomized strategy again).

- If $y_1 \text{ NOR } y_2 = 1$ then $y_1 = y_2 = 0$. We need to compute both, but the expected number of evaluations is $3/2$ in both cases, so the total is 3.
- If $y_1 \text{ NOR } y_2 = 0$ then $y_1 = 1$ or $y_2 = 1$. With probability $\geq 1/2$ we get $y_r = 1$ and we can stop. Assuming that the evaluation of the y_i is maximally expensive, the expected value is at most

$$\frac{1}{2} \cdot 2 + \frac{1}{2} \cdot 4 = 3.$$

So, in both cases the expected number is 3.

Apply the algorithm to a formula $f(x_1, \dots, x_n)$ of height $2k$. The observation shows that in each 4-way branching, we multiply the expected number of evaluations not by 4, only by 3. Thus, the expected number of evaluated variables is not $n = 4^k$, only

$$3^k = 2^{k \log 3} = 2^{2k(1/2) \log 3} = n^\alpha.$$

with $\alpha = \log \sqrt{3} < 1$.

Is this the best one can do?

- (A) We want a **lower bound** on the expected value on arbitrary randomized algorithms, on the worst inputs.
- (B) Instead we will find some (clever) input distribution and lowerbound the expected value of all deterministic algorithms on it.

The easy part of Yao's Theorem says that solving (B) gives a lower bound for (A), too. The hard part says that if we are really clever we get the best possible lower bound.

Choose a distribution P in which each $x_i = 1$ with probability p , independently. Let $p = \frac{1}{2}(3 - \sqrt{5})$, then it satisfies

$$(1 - p)^2 = p,$$

and $\mathbf{P}[Y_j = 1] = p$ also for each internal node Y_j of the NOR tree.

We need to find a lower bound on all deterministic evaluation algorithms. This is still too many algorithms.

Definition 31

Call an algorithm **depth-first** if for each subtree T , after evaluating some variables in it, the value Y_T is not found, then the next evaluated variable is also taken from T .

The following lemma helps simplificar the situation.

Lemma 32

The smallest expected value over distribution P will (also) be achieved by a depth-first algorithm.

We will not prove this lemma (we may discuss it).

Let us give a lower bound using the lemma. If $W(k)$ is the expected value for a tree of height k , then

$$\begin{aligned}W(k) &= W(k-1) + (1-p)W(k-1) = (2-p)W(k-1) \\ &= (2-p)^k = 2^{k \log(2-p)} = n^\beta\end{aligned}$$

where $\beta = \log(2-p) = \log\left(\frac{\sqrt{5}+1}{2}\right) < \log\sqrt{3} = \alpha$.

So our lower bound does not meet the upper bound. The upper bound can be achieved by a more clever, non-independent distribution (see the references in the MR book).

Polynomial identity

Given two functions $f(x), g(x)$, is it true that $f(x) = g(x)$ for **all** input values x ?

The functions may be given by a formula, or by a complicated program.

Example 33

The matrix product checking example of above can be seen as checking

$$A(Bx) = Cx$$

for all x .

As in the example, if the identity does not hold we may hope that there will be many witnesses x .

An example where mathematics can be used is when $f(x), g(x)$ are polynomials. The following fact is well-known from elementary algebra.

Proposition 34

A degree d polynomial of one variable has at most d roots.

So, if we find $P(r) = 0$ on a random r , this can only happen if r hits one of the d roots.

But, what is this good for? Checking whether a given polynomial is 0 is trivial: just check all coefficients. In the interesting applications, the polynomial has many variables, and is given only by **computing instructions**.

Example 35

$$\det(\mathbf{A}_1 x_1 + \cdots + \mathbf{A}_k x_k + \mathbf{A}_{k+1}) = 0$$

where the \mathbf{A}_i are $n \times n$ integer matrices.

This polynomial has potentially exponential size in n , but for each fixed value of (x_1, \dots, x_k) there is a polynomial algorithm of computing the determinant: Gaussian elimination. (This is **not trivial**, since rounding of fractions is not allowed.)

We need to estimate the probability of hitting a root in a multivariate polynomial.

Lemma 36 (Schwartz)

Let $p(x_1, \dots, x_m)$ be a nonzero polynomial, with each variable having degree at most d . If r_1, \dots, r_m are selected randomly from $\{1, \dots, f\}$ then the probability that $p(r_1, \dots, r_m) = 0$ is at most md/f .

Proof. Induction on m . Let $p(x_1, \dots, x_m) = p_0 + x_1 p_1 + \dots + x_1^d p_d$, where at least one of the p_i , say p_j , is not 0. Let $q(x_1) = p(x_1, r_2, \dots, r_m)$. Two cases:

$$\begin{cases} p_j(r_2, \dots, r_m) = 0 & \text{with probability } \leq (m-1)d/f, \\ q(r_1) = 0 & \text{with probability } \leq d/f. \end{cases}$$

Total is $\leq md/f$. □

Matchings in graphs

Let $G = (V, E)$ be a bipartite graph, with the edges between sets A and B , $A = \{a_1, \dots, a_n\}$, $B = \{b_1, \dots, b_n\}$. Assign to each edge $a_i b_j$ a variable x_{ij} . Matrix M :

$$m_{ij} = \begin{cases} x_{ij} & \text{if } a_i b_j \in E(G), \\ 0 & \text{otherwise.} \end{cases}$$

Theorem 37 (König)

There is a complete matching in G if and only if $\det(M)$ is not identically 0.

Proof. Consider a term in the expansion of the determinant:

$$\pm m_{1\pi(1)} m_{2\pi(2)} \cdots m_{n\pi(n)},$$

where π is a permutation of the numbers $1, \dots, n$. For this not to be 0, we need that a_i and $b_{\pi(i)}$ be connected for all i ; in other words, that $\{a_1 b_{\pi(1)}, \dots, a_n b_{\pi(n)}\}$ be a complete matching in G . In this way, if there is no complete matching in G then the determinant is identically 0. If there are complete matchings in G then to each one of them a nonzero expansion term corresponds.

These terms **do not cancel** each other (any two of these monomials contain at least two different variables), the determinant is not identically 0. □

We found a polynomial-time randomized algorithm for the matching problem in bipartite graphs.

What's the point? We mentioned it before that there are also a polynomial-time deterministic algorithms for this problem (for example via maximum flows). But there is a theoretical advantage, since determinant-computation can be **parallelized**: performed on a parallel machine in $\log^c n$ steps.

Given a network: a directed graph $G = (V, E)$ that is strongly connected. Each edge is called a **link**. There is a **message** v_i in each point i . The message needs to get to its destination $d(i)$. For simplicity, let the mapping

$$i \mapsto d(i)$$

be a permutation.

The system operates in discrete timesteps. In each step, each link can forward only one message, the others are queued up in the queue of that link. This causes **congestion delays**.

We want an **simple routing algorithm** that minimizes the time by which all messages reach their destination. The **diameter** of G is a lower bound. How much worse can congestion make it?

By simplicity, we will mean that the algorithm is **oblivious**: it assigns the a path $\rho_i = (e_1, \dots, e_k)$ (where e_j are the edges) from source i to destination $d(i)$ independently of $d(j)$ for $j \neq i$.

Theorem 38 (Nontrivial)

On a network of outdegree d and size N , every deterministic oblivious algorithm takes at least $\Omega(\sqrt{N}/d)$ steps in the worst case.

This is not impressive for a 2 dimensional grid, where the diameter is \sqrt{N} .

For networks and parallel computers, we are especially interested in graphs with small diameter.

One such example is the **hypercube**: V is the set of binary strings of length n , has $N = 2^n$ points. Two binary strings are neighbors (in both directions) if they differ only in one bit. The degree of nodes is $n = \log N$, the diameter, is also $n = \log N$. (There are also examples of graphs with constant degree and logarithmic diameter, but the hypercube is easy to analyze.)

Example 39 (A simple oblivious algorithm)

- **Bit fixing**: keep changing the first bit in which you still differ from the destination.
- There is a simple permutation causing this to take $\Omega(\sqrt{N}/n)$ steps (flip the bits around the middle of the string).

Bit fixing has an important property: if two routes depart from each other, they will **never meet again**.

Two phases.

1. Send every packet v_i to a **random intermediate** destination $\sigma(i)$.
2. Send each v_i from $\sigma(i)$ to $d(i)$.

Use the big-fixing algorithm for each phases. Why is this good? It outwits an “unknown adversary”: no matter how maliciously the destination vector $i \mapsto d(i)$ was chosen, the malice is broken by the random intermediate step.

Theorem 40

- (a) *This algorithm achieves an expected number of steps $3n$. Thus, delay n .*
- (b) *The probability of more than $14n$ steps is $\leq 1/N$.*

To prove the theorem, the following lemma is crucial, for phase 1.

Lemma 41

Let packet v follow a sequence of edges $\rho = (e_1, \dots, e_k)$. Let S be the set of packets other than v passing through at least one of the e_j . Then the delay of v on ρ is at most $|S|$.

Proof. Let $\rho = (a_0, a_1, \dots, a_k)$, $e_i = (a_{i-1}, a_i)$. Draw a 2 dimensional grid (see figure). For each packet, v , if it dwells in point a_i at time t , we show it at coordinate $(i, t - i)$. So, as long as it is on ρ in each step it either moves right (when not delayed) or moves up (when delayed). Now each horizontal line below the top one ends in a packet leaving ρ . □

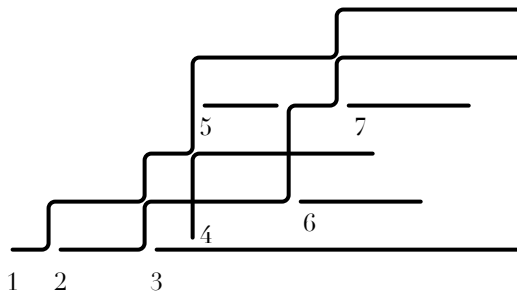


Figure: The structure of delays along a route. Going right: proceeding, going up: waiting. Messages are numbered. They can enter and leave the route, but not return.

Let $H_{ij} = 1$ if ρ_i and ρ_j share an edge, $T(e, \rho)$ the number of paths different from ρ crossing edge e , $T(e) \geq T(e, \rho)$ the number of paths crossing edge e .

We have seen that the total delay of, say, v_1 is at most

$$\sum_{j=2}^N H_{1j} \leq \sum_{e \in \rho_1} T(e, \rho_1),$$

$$\mathbf{E}\left[\sum_{j=2}^N H_{1j} \mid \rho = r\right] \leq \sum_{e \in r} \mathbf{E}T(e, r) \leq \sum_{e \in r} \mathbf{E}T(e).$$

To estimate $\mathbf{E}T(e)$:

$$\begin{aligned}\mathbf{E}|\rho_i| &= n/2, \\ \sum_e T(e) &= \sum_i |\rho_i|, \\ \sum_e \mathbf{E}T(e) &= \sum_i \mathbf{E}|\rho_i| = Nn/2.\end{aligned}$$

By symmetry, all $\mathbf{E}T(e)$ is the same for all edges e , hence

$$\begin{aligned}\sum_e \mathbf{E}T(e) &= Nn\mathbf{E}T(e) = Nn/2, \\ \mathbf{E}T(e) &= 1/2, \\ \mathbf{E}\left[\sum_{j=1}^n H_{1j} \mid \rho_1 = 4\right] &\leq |r|/2 \leq n/2.\end{aligned}$$

Hence the expected delay on $i \rightarrow \sigma(i)$ is $\leq n/2$, so the expected time is $\leq 1.5n$. The expected time for $i \rightarrow \sigma(i) \rightarrow d(i)$ is $\leq 3n$.

For a concentration result, a Chernoff bound can be used on the independent variables $H_{1j}, j = 1, \dots, N - 1$. Our final bound has to be on

$$\max_i \sum_j H_{ij},$$

so our bound on $\mathbf{P}[\sum_j H_{ij} > 14n]$ must remain small even after multiplying with N .

Counting problems: the class #P

Definition 42

Function f is in #P if there is a polynomial-time (verifier) predicate $V(x, y)$ and polynomial $p(n)$ such that for all x we have

$$f(x) = |\{y : |y| \leq p(|x|) \wedge V(x, y)\}|.$$

Reduction among #P problems. The #P-complete problems are all obviously NP-hard.

Repeated tests

How to approximate a #P function?

If y is from the range $R(x) = \Sigma^{p(n)}$, for $n = |x|$, then repeated independent tests will work only if the probability of success $\frac{f(x)}{|R(x)|}$ is not tiny. More formally, if it is not tiny compared to the standard deviation. Let X_1, \dots, X_N be i.i.d. random variables with variance σ^2 and expected value μ . Chebyshev's inequality says

$$\mathbf{P}\left[\left|\sum_i X_i/N - \mu\right| > t\sigma\right] \leq t^{-2}/N.$$

Suppose we want to estimate μ within a factor of 2, so let $t\sigma = \mu/2$, then $t = \mu/(2\sigma)$,

$$\mathbf{P}\left[\left|\sum_i X_i/N - \mu\right| > \mu/2\right] \leq (2\sigma/\mu)^2 N^{-1}$$

This will converge slowly if σ/μ is large.

Example 43

$X_i = 1$ with probability p and 0 otherwise. Then $\sigma^2 = p(1-p)$, our bound is $\frac{4(1-p)}{pN}$, and we need $N > \frac{1}{p}$ if p is small. Estimating an exponentially small probability this way would require an exponentially large number of samples.

General idea for solving this sort of problem: find a set $U(x) \subseteq R(x)$ with

- $U(x) \supseteq \{y : V(x, y) = 1\}$.
- It is easy to **generate** elements of $U(x)$.
- The quotient $\frac{|U(x)|}{f(x)}$ is “small”.

Then we need only sample the elements of $U(x)$.

(Sometimes we need a whole series $U_1(x) \supseteq U_2(x) \supseteq \dots$.)

What can we hope from a randomized algorithm?

Definition 44

An algorithm $A(x)$ is a **FPRAS** (fully polynomial randomized approximation scheme) for computing $f(x)$ if it is polynomial in $|x|$ and $1/\epsilon$, and

$$\mathbf{P}[|A(x) - f(x)| > \epsilon f(x)] \leq 1/3.$$

This definition combines **randomization** and **approximation**. We will return to the question of changing $1/3$ to arbitrary constant $\delta < 1/2$.

FPRAS for DNF satisfaction

Try to find the number of satisfying assignments of a disjunctive normal form

$$f(\mathbf{x}) = C_1(\mathbf{x}) \vee \cdots \vee C_m(\mathbf{x}),$$

where $\mathbf{x} = (x_1, \dots, x_n)$. Here each $C_i(\mathbf{x})$ is a conjunction of some variables x_k or their negations. Let $S_i = \{ \mathbf{x} : C_i(\mathbf{x}) = 1 \}$. Then it is easy to compute $|S_i|$. For example, if $C_i(\mathbf{x}) = x_1 \wedge \neg x_3 \wedge \neg x_4$ then $|S_i| = 2^{n-3}$.

We want to compute $|S|$ where $S = S_1 \cup \cdots \cup S_m$. This would be easy if the S_i were disjoint, but they are not. But we know something about the intersections. For each \mathbf{x} we can compute the **cover multiplicity** $c(\mathbf{x}) = |\{ i : \mathbf{x} \in S_i \}|$. Indeed, $c(\mathbf{x}) = |\{ i : C_i(\mathbf{x}) = 1 \}|$.

More generally, we may want to estimate the size of a set

$$S = S_1 \cup \dots \cup S_m,$$

assuming that

- We can **generate uniformly** the elements of S_i for each i (this is true for $\{x : C_i(x) = 1\}$).
- We know (can compute in polynomial time) $|S_i|$.
- For each element x , we know the **cover multiplicity** $c(x) = |\{i : x \in S_i\}|$.

Apply the idea of sampling from some small set $U(x)$: we will only sample from one of the sets S_i , but we also choose randomly from which. Knowing the cover multiplicity $c(x)$ we can compensate for the effects of overlap.

Pick $I \in \{1, \dots, m\}$ such that $\mathbf{P}[I = i] = |S_i|/M$, then pick an element $X \in S_I$ uniformly. For each x we have

$$\mathbf{P}[X = x] = \sum_{S_i \ni x} \mathbf{P}[I = i] \mathbf{P}[X = x \mid I = i] = \sum_{S_i \ni x} \frac{|S_i|}{M} \frac{1}{|S_i|} = \frac{c(x)}{M}.$$

The random variable $Y = \frac{M}{c(X)}$, has $\mathbf{E}Y = \sum_{x \in S} \frac{M}{c(x)} \mathbf{P}[X = x] = |S|$. On the other hand, $0 \leq Y \leq M$, so $\mathbf{Var}Y \leq M \leq m|S|$, therefore $\frac{\mathbf{Var}Y}{\mathbf{E}Y} \leq m$, a polynomial bound, guaranteeing that sampling Y , the law of large numbers allows to converge fast to $\mathbf{E}Y = |S|$.

Recall that an algorithm $A(x)$ is a **FPRAS** if it is polynomial in $|x|$ and $1/\varepsilon$, and

$$\mathbf{P}[|A(x) - f(x)| > \varepsilon f(x)] \leq 1/2.$$

Theorem 45

The definition of a FPRAS does not change if we replace $1/3$ on the right-hand side with any constant $\delta \leq 1/2$.

The idea of the proof is repetition, but we cannot apply the law of large numbers to $A(x)$, not knowing its expected value. We only know that it is in the interval $f(x) \pm \varepsilon f(x)$ with probability $3/4$.

Let our algorithm $B(x)$ repeat N times the algorithm $A(x)$. Let Z_1, \dots, Z_N be the values of $A(x)$ in the repetitions. Then $B(x)$ outputs the **median** M of Z_1, \dots, Z_N .

To analyze, let $U_i = 1$ if $Z_i > f(x) + \epsilon f(x)$ and 0 otherwise. Then $\mathbf{P}[U_i = 1] \leq 1/3$, while

$$\mathbf{P}[M > f(x) + \epsilon f(x)] \leq \mathbf{P}\left[\sum_i U_i \geq N/2\right] \leq e^{-N/18}$$

from the Chernoff bound. A choice $N = O(\log \frac{1}{\delta})$ makes this smaller than $\delta/2$.

We find $\mathbf{P}[M < f(x) - \epsilon f(x)] < \delta/2$ similarly.

(Following hints by Shanghua Teng.)

Let $G = (V, E)$ be an undirected graph, and $f : V \rightarrow \mathbb{R}$ a function.

We say that for some $v \in V$ the function f has a **local minimum** if $f(v) \leq f(u)$ for all neighbors u of v in G . It is an interesting algorithmic problem to look for a local minimum, where what we are trying to minimize is the **number of queries**. (Seems easier than global minimum.) Us add the a **starting point** u_0 , asking for a local minimum x with the additional condition $f(x) \leq f(u_0)$.

Consider the special case where $V = I \times J \times K$ is a lattice **slab**, where I, J, K are of the form, $I = \{i_0, i_0 + 1, \dots, i_1\}$ for integers $i_0 \leq i_1$. Two points $(x_1, y_1, z_1), (x_2, y_2, z_2)$ in V are **neighbors** if

$$|x_1 - x_2| + |y_1 - y_2| + |z_1 - z_2| \leq 1.$$

In other words, each point can only have neighbors along the coordinate axes at a distance 1.

Let $n = \max\{|I|, |J|, |K|\}$.

Theorem 46

There is an algorithm to find the minimum in time $O(n^2)$.

Proof. Without loss of generality, assume $n = |I|$. Let $m = \lfloor (i_1 + i_2)/2 \rfloor$. $I_1 = \{i_1, \dots, m\}$, $I_2 = \{m + 1, \dots, i_2\}$, $U = \{m\} \times J \times K$. Then $V = V_1 \cup V_2$ where $V_j = I_j \times J \times K$. Without loss of generality, assume $u_0 \in V_1$.

Let $f(u) = \min_{x \in U} f(x)$, for some $u \in U$ (not just local minimum). Let $u' \in V_1$, $u'' \in V_2$ be the left and right neighbors of u along dimension 1. If the minimum $f(u)$ is not unique then minimize also $f(u'')$. Cases:

- $f(u) > f(u_0)$. Search recursively in V_1 , with $u_1 = u_0$.
- $f(u) \leq f(u_0)$. Look at the two neighbors of u' , u'' of u along dimension 1. Cases:
 - $f(u) > f(u'')$. Search recursively in V_2 with $u_1 = u''$,
 - $f(u'') \geq f(u)$. Search recursively in V_1 with $u_1 = u$.

In three recursion steps, $n \rightarrow n/2$, using $< 3n^2 + 6$ queries. Hence termination in $O(n^2)$ queries. □

Theorem 47

No deterministic algorithm can do better than $O(n^2)$ in the worst case.

Here is an adversary to show it, for the cube V where $I = J = K = \{1, \dots, n\}$. For a set $S \subseteq V$ let

$$\partial(S)$$

be the outside boundary of S in G , and $d(x, y)$ the distance function in G .

The adversary will decide some answers in advance in such a way that no decided point forms a local minimum, and the undecided points form a connected component H of G . When an undecided point u will be asked it may break up H into several (at most 4) components H_i . If not then she answers with $f(u)$ smaller than in any neighbors. If yes, let H_1 be the largest of these components, and $\Delta = \partial(\cup_{i>1} H_i)$, $m = \min_{x \in \Delta} f(x)$. Let D be the maximum diameter of $H_i, i > 1$.

For each $x \in H_i$, $i > 1$ the adversary decides

$$f(x) = m - (D - d(x, u)) - 1.$$

This creates no local minimum. The lower bound rests on the following lemma.

Lemma 48

There is a time when $\frac{1}{6}|V| \leq |H| \leq \frac{5}{6}|V|$.

Proof. Let us look at the last step when $|H| > \frac{5}{6}|V|$. Then after this step we have

$$5|H_1| \geq 4|H_1| + 1 \geq |H| > \frac{5}{6}|V|.$$

$$|H_1| > \frac{1}{6}|V|.$$



The the following lemma finishes the proof.

Lemma 49

If for $H \subseteq V$ we have $\frac{1}{6}|V| \leq |H| \leq \frac{5}{6}|V|$ then $\partial(H) = \Omega(n^2)$.

This is left as exercise.

The result can be generalized to dimension d : there is a deterministic algorithm with $O(n^{d-1})$ queries, and each deterministic algorithm requires $\Omega(n^{d-1})$ queries.

The following result is completely independent of the structure of the graph. Let $|N| = |V|$, and let Δ be the maximum degree of the graph.

Theorem 50 (Aldous 83)

There is a randomized algorithm finding a local minimum in random time T with $\mathbf{E}T = O(\sqrt{N\Delta})$. Moreover, we have

$$\mathbf{P}[T/\sqrt{N\Delta} > k + 1] < e^{-k}.$$

In particular, for the d -cube we get an algorithm with $O(\sqrt{dn}^{d/2})$ expected time.

The algorithm has two stages.

1. Choose a set S of $\sqrt{N\Delta}$ points randomly. Find $f(u) = \min_{x \in S} f(x)$.
2. Follow a sequence $u = u_0, u_1, \dots, u_\tau$ where u_{i+1} is a neighbor of u_i with $f(u_{i+1}) < f(u_i)$, as long as you can. (Δ queries per move.)
When you cannot continue, you found a local minimum u_τ .

Let μ be the **rank** of the value $f(u)$ in a sorted ordering of $\{f(x) : v \in V\}$. We have

$$\begin{aligned} \mathbf{P}[\tau > k\sqrt{N/\Delta}] &\leq \mathbf{P}[\mu > k\sqrt{N/\Delta}] \leq \left(\frac{N - k\sqrt{N/\Delta}}{N}\right)^{\sqrt{N\Delta}} \\ &= \left(1 - \frac{k}{\sqrt{N\Delta}}\right)^{\sqrt{N\Delta}} \leq e^{-k}. \end{aligned}$$

This completes the proof.

The algorithm here is a primitive form of a widely used practical optimization method called **simulated annealing**. For a precise formulation, Markov processes are used. But the idea is to search for the optimum using random steps, where the size of the steps used is gradually decreased. Our example is extreme: in the first stage, we used random steps with no size constraint, in the second stage deterministic steps of minimum size.

An interesting special case is when $n = 2$, so we are searching on the vertices of a d -dimensional cube (**Hamming cube**). The upper bound is thus $\sqrt{d}2^{d/2}$. The adversary argument for the deterministic algorithm gives a lower bound $2^{d(1-\varepsilon)}$ (via a lemma similar to 49).

Can one do better with randomized algorithms?

The answer is no, but the proof is difficult. For the Hamming cube, see Aldous 83.

Recall the equality check $AB = C$ for matrices by using a random vector x . Another view of the same idea: imagine that matrices A, B are held in one location (see by player **Alice**) and the matrix C elsewhere (say by player **Bob**). Task: check the equality $AB = C$ with a minimal amount of communication. Randomized algorithm: Bob chooses a random x , sends the **fingerprint** Cx (along with x itself) to Alice, who then checks $A(Bx) = Cx$. This **communicates** only $O(n)$ bits instead of $O(n^2)$.

(In the original application we counted the operations performed: $O(n^2)$ in place of n^3 .)

Alice and Bob hold strings a and b , and want to check $a = b$ without having to communicate all of a .

Treat a, b as numbers. Choose a **random prime number** p from some set $\{p_1, \dots, p_k\}$ of primes and send $a \bmod p$. Error only if $p \mid b - a$. If a, b have n bits, then $|b - a| \leq 2^n$ has at most n prime divisors, the probability of error is bounded by n/k . For error ε choose $k = n/\varepsilon$. We need to choose from among at least k primes. Let $\pi(n)$ be the number of primes up to n . The theorem below follows from the “great” prime number theorem, but has a much simpler proof (see the Appendix of the Lovász notes).

Theorem 51 (Chebyshev)

For large n we have $\pi(n) \geq 0.75n \ln n$.

Algorithm: choose $p \leq 1.5k \ln k$. Apply a **prime test**. If p is not a prime, repeat. Else send fingerprint $a \bmod p$ ($< 2 \log n$ bits, instead of $2n$).

Bonus: From the bit string $a = a_1 a_2 \cdots a_n$, the fingerprint $a \bmod p$ can be computed with very little cost, using the following loop:

$s \leftarrow 0$.

for $i = 1$ to n :

$s \leftarrow 2s + a_i \bmod p$.

Given: universe M , with $|M| = m$, and a subset $S \subseteq M$ of size $|S| = s$, where typically s is much smaller than m . We want an **easily computable** function $h_S : M \rightarrow N$ with $|N| = n$, say $n = s^{1.1}$, in such a way that h is 1-1 on S . The set N will be called the set of **buckets**, or **bins**.

Example 52 (Application)

Decide questions of the form $x \in S$ fast, or create tables for functions $f : S \rightarrow M'$.

We may want some additional features: say, easy modification of h_S in case some elements are added to S or deleted.

In a data structures course you have learned several methods, for example via self-balancing trees. The method based on “hashing” wins in most applications by its irresistible simplicity.

Hashing starts with looking at functions $h : M \rightarrow N$ independently of S . The event $h(x) = h(y)$ for $x, y \in S$ is called a **collision**. We say then that h is a “hash function” for S if the number of collisions is small on S . In this case there will be **collision resolution methods** to create an efficient h_S . We say h is **perfect** for S if it has no collisions on S . Each function h behaves very badly on some sets S :

Theorem 53

If $m \geq s \cdot n$ then for every function $h : M \rightarrow N$ there is a set S mapped by h into a single bucket.

Proof. We have $\sum_{y \in N} |f^{-1}(y)| = m$,

$$\frac{1}{n} \sum_{y \in N} |f^{-1}(y)| = \frac{m}{n} \geq s,$$

so there is a $y_0 \in N$ with $|f^{-1}(y_0)| \geq s$. Let $S = f^{-1}(y_0)$. □

Let us look for random hash functions. For some finite set H , we can view the function

$$h : M \times H \rightarrow N$$

as a **randomized hash function**, or a **family of hash functions** in the sense that for each fixed $r \in H$ (which can be chosen randomly) the function $h(\cdot, r)$ maps from U to B .

Definition 54

The family $h(\cdot, \cdot)$ is **2-universal** if for all $x \neq y \in M$ we have

$$\mathbf{P}[h(x, r) = h(y, r)] \leq \frac{1}{n}. \quad (1)$$

This requires the probability of collisions between $h(x, r)$ and $h(y, r)$ to be smaller than between two independently and uniformly chosen random elements of N .

Example 55 (Complete independence)

$H = N^M$, and $h_1(x, (r_1, \dots, r_{|U|})) = r_x$. Here, all the values $h(x, r)$ for different x are independent.

This is useless, we want $|H|$ to be small, so that our hash function has a **small description**, thus small number of random bits.

Example 56 (Pairwise independence)

Let $p \geq m$ be a prime, $M = \{0, \dots, m-1\}$, $N = \{0, \dots, p-1\}$, and for $a, b \in \{0, \dots, p-1\}$, let

$$h_2(x, (a, b)) = ax + b \pmod{p}.$$

All the values $h_2(x, r)$ for different x are pairwise independent.

This uses few random bits, but saves nothing, since $n \geq m$. (Better solution with higher dimensions: see homework.)

Example 57 (No collision)

Restrict the previous example to $a \neq 0$, so $|H| = p(p-1)$. Fix $x \neq y$, let

$$U = h_2(x, (a, b)), \quad V = h_2(y, (a, b)).$$

We have $U \neq V$; moreover, for each $u, v \in \{0, \dots, p-1\}$, $u \neq v$ we have $\mathbf{P}[U = u, V = v] = 1/|H|$.

Example 58 (Decreasing the range)

In the previous example, let $h_3(x, (a, b)) = h_2(x, (a, b)) \bmod n$.

Theorem 59

The mapping $h_3(x, r)$ is 2-universal.

Proof. In Example 57, we have

$$\begin{aligned} \mathbf{P}[V \equiv u \pmod{n} \mid U = u] &\leq \frac{\lceil p/n \rceil - 1}{p - 1}, \\ n(\lceil p/n \rceil - 1) &< n(p/n) = p, \\ n(\lceil p/n \rceil - 1) &\leq p - 1, \\ \frac{\lceil p/n \rceil - 1}{p - 1} &\leq \frac{1}{n}. \end{aligned}$$



Why do we need b in $ax + b$ here? It seems only a ballast, but it does make things simpler. It just eliminates the problem that

$$(ax \bmod p) - (ay \bmod p) \equiv 0 \pmod{n}$$

does **not** imply

$$(a(x - y) \bmod p) \equiv 0 \pmod{n}.$$

Markov chains

Let X_0, X_1, \dots be a sequence of random variables with values in a discrete **state space** $V = \{s_1, s_2, \dots\}$ of finite or infinite size. the sequence is called a **Markov chain** if for all t , for all s_0, \dots, s_{t+1} we have

$$\mathbf{P}[X_{t+1} = s_{t+1} \mid X_0 = s_0, \dots, X_t = s_t] = \mathbf{P}[X_{t+1} = s_{t+1} \mid X_t = s_t].$$

In words, X_{t+1} depends on X_0, \dots, X_{t-1} only through X_t .

Example 60

Let Y_1, Y_2, \dots be a sequence of independent random variables, $S_0 = 0$, $S_t = Y_1 + \dots + Y_t$. Then the sequence S_0, S_1, \dots forms a Markov chain.

If for each i, j the value $p_{ij} = \mathbf{P}[X_{t+1} = i \mid X_t = j]$ is independent of t then the Markov chain is called **homogenous**. The matrix (p_{ij}) is called its **transition matrix**.

Markov chains arise in many situations when describing a system developing in time. The Markov condition says that the variables encompass enough information about the system to make sure that (the probability distribution of) its future behavior X_{t+1}, X_{t+2}, \dots is determined completely by the variable X_t describing the present state: the past influences the future only through the present.

The Markov property will typically fail if we omit some information. That is, if X_0, X_1, \dots is a Markov process, f is a function then the process defined by $Y_t = f(X_t)$ is not Markov. If a process Y_0, Y_1, \dots can be written in this form it is called a **hidden Markov** process. These are used frequently to model some processes encountered in machine learning tasks (like human speech).

We will deal almost exclusively with homogenous Markov chains on a finite state space.

If the vector $\mathbf{q}(t)$ describes the distribution of X_t , that is

$q_i^{(t)} = \mathbf{P}[X_t = i]$, then

$$\mathbf{q}(t+1) = \mathbf{q}P.$$

Since the process is homogenous the numbers

$$p_{ij}^{(t)} = \mathbf{P}[X_{t+k} = j \mid X_k = i]$$

do not depend on k , and are elements of the matrix \mathbf{P}^t , that is the t -step transition matrix is the t th power of the transition matrix \mathbf{P} .

Convergence

We are interested in the *limiting behavior* of Markov chains.

A distribution q is **stationary**, or **invariant**, or an **equilibrium** if $q = qP$.

Theorem 61

Every homogenous Markov chain has a stationary distribution.

Proof sketch. Let $q(0)$ be arbitrary distribution, $q(t) = qP^t$, and let $q(t) = \frac{1}{t} \sum_{i=1}^t q(i)$. The sequence $q(t)$ may not converge, but we can select a convergent subsequence $q(t_k)$ with limit π . Show that π is stationary. □

Let v be a state. If $q_v = 0$ in all stationary distributions then we say v is **transient**, else **persistent** (this definition is different from the one given in MR).

It is helpful to view a Markov transition matrix as a directed graph $G = (V, E, \mathbf{P})$ on the state space V . There is an edge $u \rightarrow v$ if $p_{uv} > 0$, and in this case we label the edge with p_{uv} .

Such a graph can be broken into **strongly connected components**.

Those components themselves form an acyclic graph. View the edges of this graph as going downward. We call such a component **final** if no edge leaves it (is minimal). We call a Markov chain **irreducible** if the underlying graph consists of a single strong component.

The following theorem is not hard to prove.

Theorem 62

Let the matrix \mathbf{P} describe a Markov chain.

- (a) *A state v is persistent if and only if it is in a final component.*
- (b) *There is a unique stationary distribution π if and only if there is only a single final component. In this case for each initial state $\mathbf{q}(0)$ we have*
$$\frac{1}{t} \sum_{i=1}^t \mathbf{q}(i) \rightarrow \pi.$$

A finite irreducible Markov transition matrix \mathbf{P} is called **aperiodic** if there is a t with $\mathbf{P}^t > 0$.

Theorem 63

Let \mathbf{P} be a the matrix of an irreducible Markov chain with stationary distribution $\boldsymbol{\pi}$. We have $\mathbf{q}(t) \rightarrow \boldsymbol{\pi}$ for all initial distributions $\mathbf{q}(0)$ if and only if \mathbf{P} is aperiodic.

Random walk on a graph

From a directed graph $G = (V, E)$ we can define transition probabilities as follows: from each point choose each outgoing edge with equal probability. Let A be the adjacency matrix of G and D the diagonal matrix formed from the outdegrees. Then $P = D^{-1}A$. Treat an undirected graph as a directed graph in which each edge is replaced with two directed edges. Let G be a connected undirected graph, with degrees d_i , $\text{vol } G = \sum_i d_i$, $q_i = d_i / (\text{vol } G)$. It is easy to see that distribution q is stationary. Moreover, since $DP = A$ is a symmetric matrix, In other words, for all t, i, j :

$$\pi_i p_{ij} = \pi_j p_{ji}, \text{ that is}$$

$$\mathbf{P}[X_t = i, X_{t+1} = j] = \mathbf{P}[X_t = j, X_{t+1} = i].$$

Such processes are called **reversible** since their paths have the same statistics forward in time as backward. It is easy to see that if G is connected then P is aperiodic if and only if G is not bipartite.

Examples 64

- 1 A stochastic matrix P is called **doubly stochastic** if we not only have $\sum_j p_{ij} = 1$ for all i but also $\sum_i p_{ij} = 1$ for all j . It is easy to see that then it is a square matrix (say $n \times n$), and the uniform distribution is its equilibrium.
- 2 If a stochastic matrix P is symmetric then it is doubly stochastic. As a simplest example, let

$$P = \begin{pmatrix} 1-p & p \\ p & 1-p \end{pmatrix}.$$

Speed of convergence

How fast do we approach equilibrium?

It can be very slow (exponentially) for directed graphs. Indeed, consider the following example:

$$V = \{a_0, a_1, a_2, \dots, a_n, b_0, b_1, \dots, b_n\},$$

$$E = \{(a_0, a_1), (a_1, a_2), \dots, (a_n, b_0),$$

$$(b_0, b_1), (b_1, b_2), \dots, (b_n, a_0),$$

$$(a_1, a_0), (a_2, a_0), \dots, (a_n, a_0),$$

$$(b_1, b_0), (b_2, b_0), \dots, (b_n, b_0)\}.$$

If two edges leave a point let the transition probability on them be equal. For the equilibrium π we have $\sum_i \pi_{a_i} = \sum_i \pi_{b_i}$. But if the walk starts from a_0 then it will take exponential time to reach b_0 .

To analyze convergence, we use some facts of linear algebra. Let \mathbf{M} be an $n \times n$ symmetric matrix, and let $\langle \mathbf{x}, \mathbf{y} \rangle = \mathbf{x}^T \mathbf{y}$ denote the inner product of vectors.

Theorem 65

We have the following.

- (a) There is a system of orthonormal vectors $\mathbf{u}_1, \dots, \mathbf{u}_n$ that are eigenvectors of \mathbf{M} with some real eigenvalues $\lambda_1 \geq \dots \geq \lambda_n$. For an arbitrary vector $\mathbf{v} = \sum_i c_i \mathbf{u}_i$ we have

$$\mathbf{M}\mathbf{v} = \sum_i \lambda_i c_i \mathbf{u}_i.$$

- (b) If the elements of \mathbf{M} are nonnegative then $\lambda_1 \geq 0$, further λ_1 has the largest absolute value among the λ_i , and $\mathbf{u}_1 \geq 0$.
- (c) If also \mathbf{M} is aperiodic then $\mathbf{u}_1 > 0$ and $\lambda_1 > |\lambda_i|$ for $i > 1$.

Let $\lambda = \max_{i>1} |\lambda_i|$, then for all $\mathbf{x} = c_1\mathbf{u}_1 + \cdots + c_n\mathbf{u}_n$ we have

$$|\mathbf{M}^t\mathbf{x} - c\mathbf{u}_1| = O(\lambda^t),$$

that is $\mathbf{M}^t\mathbf{x}$ converges to $c\mathbf{u}_1$ exponentially fast, namely with speed λ^t . The speed of the convergence depends on $1 - \lambda$, often called the **eigenvalue gap**.

How about nonsymmetric matrices?

If \mathbf{M} is nonsymmetric, there is a similar, but more complex estimate, based on the theory of minimal polynomials of the matrix \mathbf{M} .

Let us restrict attention to random walks on a connected graph. The matrix

$$M = Q^{1/2}PQ^{-1/2} = D^{-1/2}AD^{-1/2}$$

is symmetric.

- Frequently we construct our random walk and can add a self-loop with probability $1/2$ to each node. This results in a new transition matrix $P' = \frac{1}{2}(I + P)$, with eigenvalues $\lambda'_i = (1 + \lambda_i)/2$. All these are nonnegative, so P' is positive definite.
- If all nodes have the same degree d , then $M = P = A/d$. This simple case will be important.

Theorem 66

We have $\|Mx\| \leq \|x\|$ for all x , and hence $|\lambda_i| \leq 1$.

Proof. With $W = QP$ we have, using the Cauchy-Schwartz inequality, for any function $f, g : V \times V \rightarrow \mathbb{R}$:

$$\sum_{ij} w_{ij} f(i, j) g(i, j) \leq \left(\sum_{ij} f(i, j) w_{ij} \right)^{1/2} \left(\sum_{ij} g(i, j) w_{ij} \right)^{1/2}.$$

Apply it to $f(i, j) = u_i$, $g(i, j) = u_j$:

$$\begin{aligned} \left(\sum_i u_i q_i \right)^{1/2} \left(\sum_j v_j r_j \right)^{1/2} &\geq \sum_{ij} w_{ij} u_i v_j = \sum_i q_i u_i \sum_j p_{ij} v_j \\ &= \langle Q\mathbf{u}, P\mathbf{v} \rangle = \langle Q^{1/2}\mathbf{u}, Q^{1/2}P\mathbf{v} \rangle. \end{aligned}$$

Substituting $\mathbf{u}' = Q^{1/2}\mathbf{u}$, $\mathbf{v}' = Q^{1/2}P\mathbf{v}$:

$$\|\mathbf{u}'\| \|\mathbf{v}'\| \geq \langle \mathbf{u}', M\mathbf{v}' \rangle.$$

Setting $\mathbf{u}' = M\mathbf{v}'$ gives $\|M\mathbf{v}'\| \leq \|\mathbf{v}'\|$. □

The vector $\mathbf{g} = \mathbf{Q}^{1/2}\mathbf{1}$ is an eigenvector of M , with $\|\mathbf{g}\| = 1$ and eigenvalue 1, and it follows from Theorem 66 that this eigenvalue has maximal absolute value.

Assume that G is not bipartite: then $\lambda_1 > |\lambda_i|$ for $i > 1$.

Let $f = \sum_i c_i u_i$, then as said earlier,

$$\begin{aligned} Q^{1/2} P^t Q^{-1/2} f &= M^t f \rightarrow c_1 Q^{1/2} \mathbf{1}, \\ P^t v &\rightarrow c_1 \mathbf{1}, \end{aligned}$$

where $v = Q^{1/2} f$ is arbitrary. Similarly,

$$\begin{aligned} f^T Q^{1/2} P^t Q^{-1/2} &\rightarrow c_1 \mathbf{1}^T Q^{1/2}, \\ v^T P^t &\rightarrow c_1 \mathbf{1}^T Q = c_1 q. \end{aligned}$$

If v is a distribution then $c_1 = 1$ and we see convergence to the equilibrium q .

Here is a useful expression for the eigenvalue gap. Note that $1 - \lambda_2$ is an eigenvalue of the so-called **Laplacian** matrix

$L = I - M = D^{-1/2}(D - A)D^{-1/2}$. For $\mathbf{y} = D^{-1/2}\mathbf{x}$ we have

$$\begin{aligned}\langle \mathbf{x}, L\mathbf{x} \rangle &= \langle \mathbf{x}, D^{-1/2}(D - A)D^{-1/2}\mathbf{x} \rangle = \langle \mathbf{y}, (D - A)\mathbf{y} \rangle \\ &= \sum_i d_i y_i^2 - \sum_{j \sim i} y_i y_j = \sum_{i \sim j} (y_i - y_j)^2.\end{aligned}$$

Recall that $\mathbf{g} = D^{1/2}\mathbf{1}$ is the eigenvector of L belonging to eigenvalue 0. The second eigenvalue of L is given by the minimization:

$$\begin{aligned}1 - \lambda_2 &= \min_{\langle \mathbf{x}, D^{1/2}\mathbf{1} \rangle = 0, \|\mathbf{x}\|=1} \langle \mathbf{x}, L\mathbf{x} \rangle = \min_{\langle \mathbf{y}, D\mathbf{1} \rangle = 0, \langle \mathbf{y}, D\mathbf{y} \rangle = 1} \langle \mathbf{y}, L\mathbf{y} \rangle \\ &= \min_{\sum_i d_i y_i = 0} \frac{\sum_{i \sim j} (y_i - y_j)^2}{\sum_i d_i y_i^2}.\end{aligned}\tag{2}$$

How small can be the eigenvalue gap $1 - \lambda_2$? The following theorem follows easily from the expression 2 above:

Theorem 67

We have

$$1 - \lambda_2 \geq \frac{1}{(\text{diam } G)(\text{vol } G)}.$$

- This is inverse polynomial in $|V|$, showing that the convergence is not worse than polynomial in $|V|$. This is much better than what we had in directed graphs.
- However, in our typical applications the Markov chain in question will have an exponential number of states, hence we will need an eigenvalue gap that has an inverse polynomial lower bound in $\log |V|$.

Monte-Carlo sampling applications

In typical applications a distribution q is given over a set V that with **exponential size**. Probabilities of individual elements might be given (the distribution q may even be uniform). But to approximate the probabilities of certain interesting sets, we need to sample from q .

Example 68 (Ising model)

Call two pairs $(i, j), (i', j')$ are **neighbors** and write $(i, j) \sim (i', j')$ if $|i - i'| + |j - j'| = 1$. View a matrix σ with elements in $\{-1, 1\}$ like a “ferromagnet” with individual atomic magnets pointing up or down. The **energy** $U(\sigma)$ is defined as

$$- \sum_{u \sim v} \sigma_u \sigma_v.$$

Let $\beta > 0$ be a fixed parameter (the “inverse temperature”). We define $Z_\beta = \sum_{\sigma} e^{-\beta U(\sigma)}$, and the probability

$$q_{\sigma}(\beta) = e^{-\beta U(\sigma)} / Z(\beta)$$

(the so-called Boltzmann distribution). Many versions of this example are of central interest in physics.

An appropriately designed Markov chain will converge to q and allow sampling from it.

Designing a reversible Markov chain

Suppose that a graph $G = (V, E)$ is given, along with a distribution q over V . We need to define transition probabilities p_{uv} along the edges for which q is a reversible distribution:

$$q_u p_{uv} = q_v p_{vu}.$$

This is always satisfied for $u = v$, so we can use p_{uu} for balance in $\sum_v p_{uv} = 1$. Also, the relation remains true if we multiply p_{uv} by c_{uv} where $c_{uv} = c_{vu}$. Let $C = (c_{uv})$ be an arbitrary symmetric stochastic matrix (say $c_{uv} = 1/d$ for $u \neq v$ and maximum degree d). The following formula, called the **Metropolis algorithm**, is a popular choice:

$$p_{uv} = c_{uv} \min \left(1, \frac{q_v}{q_u} \right) \text{ for } v \neq u.$$

Check that this is reversible!

Example 69 (Subgraph)

Let $G = (V, E)$ where $V = \{1, \dots, n\} \times \{1, \dots, n\}$, $(i, j) \sim (i', j')$ if $|i - i'| + |j - j'| = 1$. Let q be the uniform distribution. The transitions for the Metropolis algorithm give

$$p_{uv} = \frac{1}{4} \min \left(1, \frac{q_v}{q_u} \right) \text{ for } v \neq u.$$

In words: try to move in each of the 4 directions (north, south, east, west) with probability $1/4$. If the move would take you outside the square, stay in place.

Notice that (on the border) this is not the same as the random walk on the graph. That random walk would give smaller stationary probability to the points on the border.

Example 70 (Metropolis algorithm for the Ising model)

For states σ, σ' of the Ising model say $\sigma \sim \sigma'$ if they differ only in a single position $u = (i, j)$, that is $\sigma_u \neq \sigma'_u$. In this case

$$\frac{q_{\sigma'}}{q_{\sigma}} = e^{\beta(U(\sigma) - U(\sigma'))}$$

is easy to compute, depends only the values of σ_v for neighbors v of lattice point u , so it gives rise to a Markov process also called a **probabilistic cellular automaton**.

In other applications, we are interested in the approximate sizes of two sets, $A \subseteq B \subseteq \{0, 1\}^n$ decidable in polynomial time. If $|B|/2^n$ is exponentially small then we cannot just sample using random elements of $\{0, 1\}^n$. But we could find out $|A|/|B|$ sampling uniformly from B .

How to sample uniformly from B ? We might find a Markov chain in B converging fast to its equilibrium distribution, and make sure that the latter is uniform on B .

Example 71 (Counting matchings)

Let M_k be the set of matchings of size k in a bipartite graph G . Under certain conditions one can estimate M_k for each k . For this, one will estimate

$$\frac{|M_k|}{|M_{k-1} \cup M_k|}$$

for each k starting from $k = 1$.

We sample from a Markov chain on the set $M_{k-1} \cup M_k$. For this, we define a graph whose points are matchings in $M_{k-1} \cup M_k$. Let m, m' be two matchings $m, m' \in M_k \cup M_{k-1}$. We say that they are **neighbors** if there is an $e = (u, v) \in V^2$ such that one of the following holds:

Addition $e \in E, m \in M_{k-1}, m' \in M_k, m' = m \cup \{e\}$.

Deletion $e \in E, m' \in M_{k-1}, m \in M_k, m = m' \cup \{e\}$.

Rotation $m, m' \in M_{k-1}$ and there is a w with $(u, w) \in m$ and $(v, w) \in m'$.

Now the Markov chain can be defined by the Metropolis algorithm: choose a random (u, v) apply an operation determined by (u, v) if it is possible, else do nothing. Irreducibility is easy to establish. The difficulty is in estimating its speed of convergence (mixing), that is the eigenvalue gap.

Estimating the eigenvalue gap

We have seen that the speed of convergence (say the number of steps needed to get within distance ε of the equilibrium) is measured by the eigenvalue gap and is crucial. Here is a spectrum of possibilities:

- Directed graphs: it is possible to get **exponential**-time convergence.
- Undirected graphs: **polynomial**-time convergence.
- Sampling, approximate counting: we need **logarithmic** convergence as a function of the number of states, since the number of states is typically exponential in the size of our computer.
- Expanders (see later): **constant**-time convergence.

We define a quantity that is often helpful in estimating the eigenvalue gap. For any $S \subset V$ let $\delta(S)$ be the set of edges between S and $V \setminus S$,

$$q(S) = \sum_{i \in S} q_i, \quad q(\delta(S)) = \sum_{(i,j) \in \delta(S)} q_i p_{ij},$$

$$\Phi(S) = \frac{q(\delta(S))}{q(S)q(V \setminus S)}, \quad \Phi = \min_S \Phi(S).$$

This quantity, called the **conductance** tries to minimize the **relative** probability flow between two parts of the graph. The numerator is the probability of seeing an element of S followed by that of $V \setminus S$ in the random walk, the denominator is the probability of seeing this in independent sampling according to q .

- The following simpler variant is easier to interpret:

$$\Phi' = \min_{q(S) \leq 1/2} \frac{q(\delta(S))}{q(S)}. \text{ Clearly } \Phi' \leq \Phi \leq 2\Phi'.$$

- If the process is a graph random walk, then $q(\delta(S)) = |S|/(\text{vol } G)$.
- If the graph is regular, then $\text{vol } G = dn$, $q(S) = |S|/n$,

$$\Phi(S) = \frac{n}{d} \frac{|\delta(S)|}{|S| \cdot |V \setminus S|}, \quad \Phi' = \min_{|S| \leq n/2} \frac{|\delta(S)|}{d|S|}.$$

Theorem 72

We have $\Phi^2/8 \leq 1 - \lambda_2 \leq \Phi$.

The upper bound follows easily by choosing y_i to be constant on S and on $V \setminus S$. The lower bound is harder, we will take it on faith.

How to lowerbound the conductance? The following estimate has been successful.

Theorem 73

Let c be a capacity defined on all edges. Suppose that for each $i \neq j$ we can construct a flow f_{ij} of value $q_i q_j$ from i to j , in such a way that the sum of all these flows on any edge is at most c . Then $\Phi \geq \frac{1}{c(\text{vol } G)}$.

Proof. Let $\Phi = \Phi(S)$. The total amount of flow from S to $V \setminus S$ is $q(S)q(V \setminus S)$. Summing these up on the edges of the cut $\delta(S)$, we get

$$q(S)q(V \setminus S) \leq |\delta(S)|c, \quad \frac{1}{c(\text{vol } G)} \leq \frac{q(\delta(S))}{q(S)q(V \setminus S)} = \Phi(S).$$



Your book illustrates this technique on the example of the Markov chain on matchings.

An “expander” is essentially a graph whose eigenvalue gap is lowerbounded by some constant.

- The definition then depends on the constant. Typically we look at an **infinite family** of expanders, all sharing the same constant.
- We will restrict attention to bipartite, regular graphs, but allowing **multigraphs**: edges have integer multiplicity.

The eigenvalue gap measures, in a way, the **degree of connectivity**, as seen from the formula

$$\Phi' = \min_{|S| \leq n/2} \frac{|\delta(S)|}{d|S|}.$$

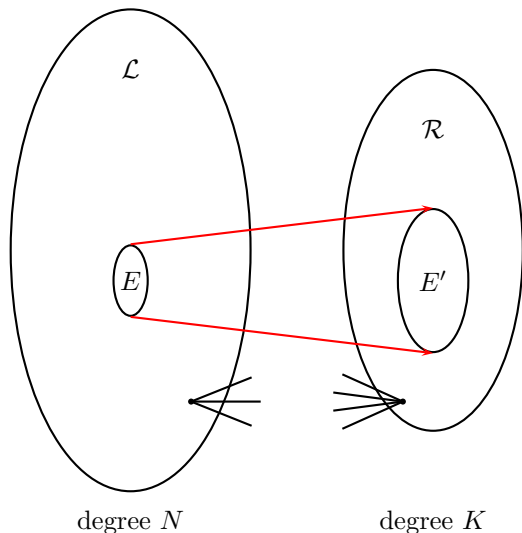
seen earlier. This says that for any set S of size $\leq n/2$, at least a fraction Φ' of all edges starting from points of S must go to $V \setminus S$.

Nodes of a bipartite graph will be distinguished as **left** and **right** nodes. For a subset S of the left set of $G = (V, E)$, let $\text{Nb}(S)$ consist of the points connected by some edge to some element of S : the **neighbors** of S . Consider a d -regular bipartite graph G , with the left and right sets having n nodes. We say that the graph **expands** S by a factor λ if we have $|\text{Nb}(S)| \geq \lambda|S|$.

Definition 74

For $\alpha, \lambda > 0$, a graph G is a (d, α, λ, n) -**expander** if it expands every subset S of size $\leq \alpha n$ of the left set by a factor λ .

An expander. Here, the degree of points is different in the left set and in the right set.



Random expanders

Are there expanders with constant d, α, λ and arbitrarily large n ? There is a simple proof that can be seen as the illustration of the **probabilistic method**. We will choose a random bipartite multigraph of degree d and show that it is expander with positive probability. Choosing α small (constant) can go λ arbitrarily close to d (you cannot hope better).

The random choice

Start with dn left nodes u_1, \dots, u_{dn} and dn right nodes v_1, \dots, v_{dn} . Choose a random complete matching among these. Call the resulting graph M . Obtain G as follows: collapse each group of d left nodes into a single node: u_1, \dots, u_d into one node, u_{d+1}, \dots, u_{2d} into another node, and so on. Similarly collapse each group of d right nodes. Edges are inherited from the ancestors. The process may give multiple edges: a multigraph B . Two nodes of M are called **cluster neighbors** if they are collapsed to the same node of B .

Theorem 75

For each $f > 0$, if α is sufficiently small then this process gives a $(d, \alpha, d(1 - f), n)$ -expander with positive probability for all sufficiently large n .

In what follows we will prove this.

Let S be a set of size αn in the left set of G . We will estimate the probability that E has too few neighbors. In the above choice of the graph G we might as well start with assigning edges to the nodes of S , in some fixed order of the preimage of S in M . Call a node of the right set of M **occupied** if it has a cluster neighbor already reached by an earlier edge. Let random variable X_i be 1 if the i th edge goes to an occupied node and 0 otherwise. There are

$$dn - i + 1 \geq dn - d\alpha n = dn(1 - \alpha)$$

choices for the i th edge, at most $d^2|S|$ of these are occupied. Therefore

$$\mathbf{P}[X_i = 1 \mid X_1, \dots, X_{i-1}] \leq \frac{d^2|S|}{dn(1 - \alpha)} = \frac{d\alpha}{1 - \alpha} =: p.$$

Using a generalization of the large deviations theorem (“Chernoff bound”) we have, for $f > 0$:

$$\mathbf{P}\left[\sum_{i=1}^{d\alpha n} X_i \geq fd\alpha n\right] \leq e^{-d\alpha n D(f,p)} \leq \left(\frac{ep}{f}\right)^{fd\alpha n}.$$

Now, the number of different neighbors of S is $d\alpha n - \sum_i X_i$, hence

$$\mathbf{P}[\text{Nb}(S) \leq d\alpha n(1-f)] \leq \left(\frac{ep}{f}\right)^{fd\alpha n} = \left(\frac{ed\alpha}{f(1-\alpha)}\right)^{fd\alpha n}.$$

Multiply with the number of sets S of size $\leq \alpha n$:

$$\sum_{i \leq \alpha n} \binom{n}{\alpha n} \leq \left(\frac{e}{\alpha}\right)^{\alpha n}.$$

(The inequality was proved earlier.) We get

$$\left(\frac{e}{\alpha}\right)^{\alpha n} \left(\frac{e d \alpha}{f(1-\alpha)}\right)^{f d \alpha n} = \left(\frac{e^2 d}{f(1-\alpha)} \left(\frac{e \alpha d}{f(1-\alpha)}\right)^{f d - 1}\right)^{\alpha n}.$$

The base is < 1 if $f > 1/d$ and α is sufficiently small.

The above proof is an **existence proof**.

- Not only does not it help us compute an expander efficiently, but even if we are handed one, does not give any effective way of checking it.
- However, if we just want a large eigenvalue gap, that can be computed effectively.
- In many theoretical applications, we need efficiently computable expanders: say, of size 2^n in which the neighbors of each point are listed in time polynomial in n . Such constructions exist. In this course, we will not see them, since the proofs are generally complicated. But we will see an application.

BPP amplification

Let $L \in \text{BPP}$, with a randomized polynomial algorithm $A(x, r)$, that for each x decides whether it is in L , and fails only with probability $\leq 1/100$. Assume $|r| = |x| = n$. How to decrease the error probability to, say, 2^{-k} ? We learned to repeat independently some number of times and to take the majority. In some cases, we must be parsimonious with the **number of random bits** used. Two interesting possibilities:

- 1 Use independent repetitions r_1, \dots, r_m . The Chernoff bound shows that $O(nk)$ random bits suffice, with and the number of operations still polynomial in n, k .
- 2 Use pairwise independent random strings of the form $r_i = ai + b \pmod{p}$: then $O(n + k)$ random bits suffice, but the number of operations becomes exponential in k .

With constructive expanders, we will only use $n + O(k)$ random bits and still a polynomial number of operations.

Algorithm 1 (Call it $B(x, r)$.)

Let $N = 2^n$. For some constants d, δ , take a d -regular undirected graph $G_N = (V_N, E_N)$ for $V_N = \{1, 2, \dots, N\}$. With self-loops, we also achieve that λ_2 is the second largest eigenvalue. Assume $\delta \leq 1 - \lambda_2$, so G is an “expander”. The graph must be given in such a way that for each point i , the list of its d neighbors is **computable** in time polynomial in n .

Choose a random $r_0 = R(0) \in V_N$ uniformly. Let $R(0), R(1), R(2), \dots$ be a random walk over G_N . Given $R(i-1)$, we need only a new random binary string r_i of size $\lceil \log d \rceil$ to find $R(i+1)$, since we only need to choose between the outgoing edges.

Let β be such that $\lambda_2^\beta \leq 0.1$. Compute $A(x, R(i\beta))$ for each $i = 1, \dots, m$ and take the majority.

We will achieve $\mathbf{P}[B(x, r) \text{ is wrong}] \leq 2^{-k}$ with $m = O(k)$.

Let $X_1, X_2, \dots \in \{0, 1\}$ be independent variables with $\mathbf{P}[X_i = 1] \leq \sigma$, and let $i_1 < i_2 < \dots < i_k$. Then $\mathbf{P}[X_{i_1} = 1, \dots, X_{i_k} = 1] \leq \sigma^{-k}$. We will develop a similar estimate for our Markov chain.

Let \mathbf{P} be transition matrix obtained by making β steps of the random walk on our expander. Then $\lambda_2(\mathbf{P}) \leq 0.1$. The equilibrium distribution π is still the uniform distribution.

For a vector \mathbf{x} define $\|\mathbf{x}\|_1 = \sum_i |x_i|$,
 $\|\mathbf{x}\| = \|\mathbf{x}\|_2 = (\sum_i x_i^2)^{1/2} = \langle \mathbf{x}, \mathbf{x} \rangle^{1/2}$, then the Cauchy-Schwartz inequality gives

$$\|\mathbf{x}\|_1 \leq \sqrt{N} \|\mathbf{x}\|.$$

For a matrix A let $\|A\| = \|A\|_2 = \min_{\mathbf{x} \neq \mathbf{0}} \|\mathbf{x}A\| / \|\mathbf{x}\|$. It is easy to see $\|AB\| \leq \|A\| \|B\|$.

Let A be a symmetric matrix with eigenvalues λ_i , then it is easy to see

$$\|A\| = \max_i |\lambda_i|.$$

Let $X_i = R(i\beta)$, then X_i is a random walk with matrix P and initial distribution q . For a set $S \in V$, let D_S be the diagonal matrix with 1's in positions $i \in S$ and 0's elsewhere. We have

$$\mathbf{P}[X_1 \in S_1, \dots, X_m \in S_m] = \|qPD_{S_1} \cdots PD_{S_m}\|_1 \leq \sqrt{N} \|qPD_{S_1} \cdots PD_{S_m}\|.$$

Though our final interest is in the norm $\|\cdot\|_1$, due to the better properties of the norm $\|\cdot\| = \|\cdot\|_2$ we will go through estimating $\|PD_{S_1} \cdots PD_{S_m}\| = \|PD_{S_1}\| \cdots \|PD_{S_m}\|$.

Lemma 76

Let S be a set of states with $\pi(S) \leq \sigma$. We have

$$\|PD_S\| \leq \min(1, \sigma^{1/2} + \lambda_2).$$

Proof. Let π, u_2, \dots, u_n be an orthonormal basis of row eigenvectors of P . For any vector $q = c_1\pi + \sum_i c_i u_i =: u + v$ we have

$$qP = u + \sum_{i>1} \lambda_i c_i u_i =: u + v',$$

$$\|v'\| \leq \lambda_2 \|v\|,$$

$$\|qPD_S\| \leq \|qP\| \leq \|q\|,$$

$$\|qPD_S\| \leq \|uD_S\| + \|v'D_S\|$$

$$\leq \sigma^{1/2} \|u\| + \lambda_2 \|v\| \leq (\sigma^{1/2} + \lambda_2) \|q\|.$$



Let $S = \{r : A(x, r) \text{ is wrong}\}$, and assume $\pi(S) \leq 0.01$. Using $\lambda_2 \leq 0.1$ and lemma 76 gives $\|PD_S\| \leq 0.1 + 0.1 = 1/5$.

Let S_1, S_2, \dots, S_m be a sequence of sets where each S_i is either S or $V_N \setminus S$ and there are w occurrences of S . Then we obtain

$$\begin{aligned} \|\pi\| &= N^{-1/2}, \\ \|PD_{S_1} \cdots PD_{S_m}\| &\leq (1/5)^m (4/5)^{m-w} \leq (1/5)^w, \\ \|\pi PD_{S_1} \cdots PD_{S_m}\| &\leq \|\pi\| \|PD_{S_1} \cdots PD_{S_m}\| \leq N^{-1/2} (1/5)^w, \\ \|\pi PD_{S_1} \cdots PD_{S_m}\|_1 &\leq N^{1/2} N^{-1/2} (1/5)^w = (1/5)^w. \end{aligned}$$

A wrong majority decision corresponds to a sequence S_1, \dots, S_m with at least $w \geq m/2$ occurrences of S . There are at most 2^m different ways to choose this sequence, therefore the probability of a wrong decision is at most $2^m (1/5)^{m/2} = (4/5)^{m/2}$.

What is probability?

What is probability? The question is often answered referring to relative frequency in repeated experiments.

- The law of large numbers offers an “internal” justification of this interpretation.
- However, the Markov chain applications seen here go beyond this interpretation. The BPP amplification does not repeat experiments and achieves a very small probability of error. Repeating the experiment independently would be a waste of effort defeating the purpose of the algorithm.
- Markov-chain Monte-Carlo works on a space of exponential size, but takes a sample in a polynomial number of steps that has nearly the desired distribution. Each sample will be different, making a frequentist interpretation again dubious.

Randomness

- Probability theory started approximately with Pascal (mid 17th century).
- Its current mathematical framework was introduced by Kolmogorov (1931). This framework does not address the question: **what is a random string?**
- This question was raised maybe first by Laplace (early 19th century), then by von Mises (early 20th century). Von Mises proposed the convergence of relative frequency on subsequences generated by some “rule” as a criterion. But he could not formalize the notion of a rule.
- Church (1947) defined “rule” as a recursive rule. But the frequency test proved insufficient, as shown by Ville.
- The current theory is based on Kolmogorov’s introduction of description complexity (1965) and later work by Martin-Löf and Levin.

The paradox giving rise to the notion of randomness is this. Suppose your friend gives you a 0-1 sequence $x_1x_2 \cdots x_{100}$ of length 100, and told that it is the result of a series of coin tosses he made the day before. If the sequence is 010101 \cdots 01 then you will not believe your friend. He may challenge you for your reasons, however, since each sequence of length 100 has the same probability 2^{-100} .

You may say, “yes, but this sequence is created by a rule”, but **what is a rule**? There is an infinite number of possible rules.

Laplace’s guessed that there are only few sequences obeying a **simple rule**, but without saying what a simple rule is. Kolmogorov formalized the notion of the simplicity needed here.

Description complexity

Fix some alphabet $\Sigma \supset \{0, 1\}$. Let T be a Turing machine with

- input tape with alphabet $\{0, 1, *\}$, where $*$ is the “blank symbol”,
- output tape with alphabet $\Sigma \cup \{*\}$.
- work tape with some alphabet containing $*$.

We define the **partial** function $T : \{0, 1\}^* \rightarrow \Sigma^*$ as follows. To compute $T(p)$ we write string p onto the input tape of T , leave the other tapes blank. We start T . If T halts and the output tape contains a single nonblank string x at the beginning, then $T(p) = x$. Otherwise $T(p)$ is not defined.

We view the machine T as an **interpreter** of descriptions. Thus, we say that the binary string p **describes** the output $T(p)$. (We use binary descriptions for having a common base of comparison.) Let

$$K_T(x) = \min_{T(p)=x} |p|.$$

We say that $K_T(x)$ is the **description complexity**, or Kolmogorov complexity, of string x on machine T . This notion is, of course, machine-dependent. For every string x there is an interpreter T_x such that $K_{T_x}(x) = 0$. But interestingly, the machine-dependence is quite moderate.

Theorem 77 (Invariance)

*There is an interpreter U that is **optimal** in the following sense. For every other interpreter T there is a constant c_T such that for all strings $x \in \Sigma^*$ we have*

$$K_U(x) \leq K_T(x) + c_T.$$

The theorem shows that no other interpreter T can have much shorter descriptions than the optimal interpreter U , since the difference will be bounded by a constant (dependent on T).

Proof. There is an interpreter (an appropriate universal Turing machine) U such that for all interpreters T there is a string q_T such that for all strings $p \in \{0, 1\}^*$ we have

$$T(p) = U(q_T p).$$

Here, q_T contains an encoded description of the machine T (its transition table). Now for all strings x and all machines T we have

$$K_U(x) \leq K_T(x) + |q_T|.$$



Fix an optimal interpreter U and write

$$K(x) = K_U(x).$$

We will use the notation $f(n) \stackrel{+}{<} g(n)$ to mean $f(n) \leq g(n) + O(1)$. The notion $f(n) \stackrel{\pm}{=} g(n)$ is defined similarly.

The function $K(x)$ has several synonymous names, some of them suggest other interpretations.

- Description complexity.
- Minimal compression size.
- Amount of individual information.
- Algorithmic information.
- Algorithmic entropy.

Upper and lower bound

Theorem 78

For a binary string x of length n we have $K(x) \stackrel{+}{<} n$.

Proof. Define a machine T that simply outputs its input. Then apply the invariance theorem. \square

The following theorem shows that this bound is sharp, in a statistical sense.

Theorem 79

Let X be a uniformly chosen random binary string of length n . Then

$$\mathbf{P}[K(X) < n - k] < 2^{-k}.$$

Proof. There are at most 2^i descriptions of length i , so at most $1 + 2 + 4 + \dots + 2^{n-k-1} < 2^{n-k}$ strings with complexity $< n - k$. \square

Example 80

Note that if the string x is of the form $0101 \cdots 01$ ($n/2$ times) then

$$K(x) \stackrel{+}{<} \log n.$$

Since as we have seen, the probability is very small of getting **any** string of such low-complexity, we have a sort of justification for our suspicion about its coin-tossing pedigree. In fact, a good argument can be made (in the more advanced versions of this theory) to view $n - K(x)$ as a measure of the **non-randomness** of string x .

Example 81

If x is a binary string of length n such that $\sum_i x_i = k$ then with $p = k/n$ we have

$$K(x) \stackrel{+}{\leq} \log \binom{n}{k} + O(\log n) = nH(p) + O(\log n),$$

where $H(p) = -p \log p - (1-p) \log(1-p)$. (This indicates a relation to information.)

Indeed, given n and k (by strings of length $\log n$) we can enumerate all binary strings containing k 1's, and describe x by the rank of x in this enumeration.

The following theorem limits the usefulness of the function $K(x)$.

Theorem 82

The function $K(x)$ is not computable.

Proof. (By contradiction.) Assume that $K(x)$ is computable. For each k let $f(k)$ be the smallest x (lexicographically) with $K(x) > k$, and let T be a Turing machine computing $f(k)$ from a binary representation of k . We have

$$K(f(k)) \leq K_T(f(k)) + c_T \leq \log k + 1 + c_T.$$

But by definition we have $k < K(f(k))$, leading to a contradiction for large k . □

This proof formalizes the paradox: “the smallest number definable with fewer than 100 characters”.

Despite its non-computability, the notion of description complexity and the definitions of randomness and information built on it have proved of great value in clarifying some important issues of randomness, information and prediction.

In randomized computations and in cryptographical applications we frequently work with strings that are not random, only pseudo-random; again, in order to understand the difference, it is useful to know what randomness means.

Derandomization

Conditional expectations

Frequently, a randomized algorithm can help us to a deterministic one. We will illustrate this on the set balancing problem, which we recall. Given an $n \times m$ matrix A with 0-1 entries. We are looking for a vector b with 1, -1 entries for which

$$\|Ab\|_{\infty}$$

is minimal.

A good vector can be found by random choice. Let β_1, \dots, β_n be independent, with $\mathbf{P}[\beta_j = 1] = \mathbf{P}[\beta_j = -1] = 1/2$, and let $\boldsymbol{\beta} = (\beta_1, \dots, \beta_n)$. Let \mathbf{a}_i^T the i th row, let \mathcal{E}_i be the event $|\mathbf{a}_i^T \boldsymbol{\beta}| > 8\sqrt{m \ln n}$. We showed

$$\mathbf{P}[\mathcal{E}_i] \leq 2/n^2.$$

So with probability $1 - 2/n$, all n rows have discrepancy $\leq 8\sqrt{m \ln n}$. Let the random variable X_i be 1 if \mathcal{E}_i holds and 0 otherwise. Let $Y = \sum_i X_i$. Our theorem shows $\mathbf{E}Y \leq 2/n$.

Consider a tree with all possible values $b_1 \cdots b_n$ of the sequence $\beta_1 \cdots \beta_n$ at the leaves, and the inner nodes at level k steps down from the top labeled by values of $b_1 \cdots b_k$. If you think into it you will see that for each $b_1 \cdots b_k$, the value

$$f(b_1 \cdots b_k) = \mathbf{E}[Y \mid \beta_1 = b_1, \dots, \beta_k = b_k]$$

is computable in polynomial time, and

$$f(b_1 \cdots b_k) = (f(b_1 \cdots b_k 0) + f(b_1 \cdots b_k 1))/2.$$

Therefore the following algorithm works for computing a sequence $b_1 \cdots b_n$ recursively: For $k = 0, \dots, n - 1$, compute the b_k such that

$$f(b_1 \cdots b_{k-1} b_k) \leq f(b_1 \cdots b_{k-1} (1 - b_k)).$$

With this sequence we will have $\|\mathbf{A}\mathbf{b}\|_\infty \leq 8\sqrt{m \ln n}$. Indeed, we have

$$\mathbf{E}[Y \mid \beta_1 = b_1, \dots, \beta_n = b_n] \leq 2/n < 1.$$

But this is a deterministic integer value: if its expectation is < 1 then it is 0: in other words, none of the events \mathcal{E}_i happens.

Two-point sampling

- The following example is from the area of **parallel computing**. We have had such an example earlier, since the randomized algorithm to decide whether a graph has a matching adds value only in the parallel computing model.
- We will not give a formal definition of a parallel computer. Different versions exist (EREW, CREW, etc.). The class NC of functions computable in polylog time and with a polynomial number of processors is insensitive to these differences.

Given a graph $G = (E, V)$, we want to find a **maximal** independent set (an independent set not contained in any other independent set). (Finding a **maximum** independent set is NP-complete.) It is easy to find one in polynomial time, but the greedy algorithm of finding one is sequential. The parallel algorithm of Luby given below puts the problem into NC.

Plan:

- ① Find a randomized parallel algorithm (putting the problem in **RNC**, randomized NC).
- ② Derandomize it.

We will follow the scheme only approximately.

Definition 83

For a subset $S \subseteq V$ of the graph G , let $E(S)$ be the set of edges of G with at least one end in S .

Let $\Gamma(v)$ be the set of neighbors of point v , $d_v = |\Gamma(v)|$ its degree, and $\Gamma S = \bigcup_{v \in S} \Gamma(v)$.

Theorem 84

There is a constant c and a randomized parallel algorithm to find an independent set $S(r) \subseteq V$ with $\mathbf{E}|E(\Gamma S(r))| \geq c|E|$.

Iterating this algorithm appropriately a logarithmic number of times gives a maximal independent set.

Algorithm 2

We construct the set S as follows.

- 1 (in parallel) Put each point $v \in V$ with probability $\frac{1}{2d_v}$ into a set S_1 . Assume first that the choice is made independently for each point.
- 2 (in parallel) If an edge has both ends in S_1 then delete the lower-degree end from S_1 (break ties arbitrarily), resulting in the desired set S .

The following probabilistic analysis lowerbounds the expected value $\mathbf{E}|E(\Gamma S)|$.

Definition 85

A point is called **good** if it has at least $d_v/3$ neighbors with degree $\leq d_v$; otherwise it is bad. An edge is **good**, if at least one of its endpoints is good.

The following lemma allows us to concentrate on good edges:

Lemma 86

At least of half of all edges are good.

Proof. Let us direct the edges from lower to higher degree, and let d_v^+, d_v^- denote the in- and outdegrees. Let V_B, V_G be the bad and good points, $e(X, Y)$ be number of edges between sets X, Y .

In a bad point $d_v \leq 3(d_v^+ - d_v^-)$. Hence the sum of degrees of bad points is $\leq 3 \times$ (the flow from bad points to good points):

$$\begin{aligned} 2e(V_B, V_B) + e(V_B, V_G) + e(V_B, V_G) &\leq 3(e(V_B, V_G) - e(V_G, V_B)) \\ &\leq 3(e(V_B, V_G) + e(V_G, V_B)), \\ e(V_B, V_B) &\leq e(V_B, V_G) + e(V_G, V_B). \end{aligned}$$



Let $c_1 = 1 - e^{-1/6}$.

Lemma 87

If v is a good point with positive degree then it is ΓS_1 with probability $\geq c_1$.

Lemma 88

Every point $v \in S_1$ will also be in S with probability $\geq \frac{1}{2}$.

Lemma 86 says that at least half of the edges are good. Lemma 87 implies that a good edge will get an endpoint into ΓS_1 with probability $\geq c_1$. Lemma 88 implies that this endpoint will be also in ΓS with probability $\geq \frac{1}{2}$. Multiplying the results lowerbounds the expected value:

$$\mathbf{E}|E(\Gamma S)| \geq \frac{1}{2} \cdot c_1 \cdot \frac{1}{2} = c_1/4.$$

The theorem follows with $c = c_1/4$.

Proof of Lemma 87. We make at least $d_v/3$ independent attempts with probability $\geq \frac{1}{2d_v}$, so the the probability of not succeeding even once is $\leq (1 - \frac{1}{2d_v})^{d_v/3} \leq e^{-1/6}$. □

Proof of Lemma 88. A point v will only be deleted if a neighbor of degree $\geq d_v$ has been selected. For each such neighbor this happens with probability $\leq \frac{1}{2d_v}$. Apply the union bound to the at most d_v such neighbors. □

Derandomization

- We used n independent random variables $X_v(r)$ where $X_v(r) = 1$ if point v is selected, and 0 otherwise, and $\mathbf{P}[X(r)_v = 1] = \frac{1}{2d_v}$. The only lemma in which we used independence was Lemma 87.
- In Problem 4b of homework 2 we have already seen that with a different constant c_1 the lemma will also hold when only pairwise independence is used.
- Two-point sampling creates p independent random variables $Y_v(r)$ ranging over $\{0, \dots, p-1\}$, using only $2 \log p$ independent random bits. Using an appropriate p (say $p = O(n^2)$) this can be used to generate pairwise independent random variables $X_v(r)$, with $\mathbf{P}[X_v(r) = 1] \approx \frac{1}{2d_v}$.
- We still have $\mathbf{E}|E(\Gamma S(r))| \geq c|E|$, so there is a particular choice r_0 of these bits giving $|E(\Gamma S(r_0))| \geq c|E|$. Since now $|r| = O(\log n)$, we can search for r_0 in polynomial time.