# Probability and Computing, Oxford 2021-22

Lecturer: Leslie Ann Goldberg

9th March, 2022

CONTENTS

PART 1
INTRODUCTION TO RANDOMISED ALGORITHMS

## 1.1 A randomised algorithm for string equality

### 1.1.1 Why randomised algorithms?

Randomised algorithms make random choices. How can this help? The simple reason is that for a well-designed randomised algorithm, it is highly improbable that the algorithm will make all the wrong choices during its execution.

The advantages of randomised algorithms are:

**simplicity** randomised algorithms are usually simpler than their deterministic counterparts

**efficiency** they are faster, use less memory, or less communication than the best known deterministic algorithm

**dealing with incomplete information** randomised algorithms are better in adversarial situations

Although there is a general consensus that randomised algorithms have many advantages over their deterministic counterparts, it can be difficult to *prove* that equally-good deterministic algorithms don't exist. A few cases where this has been done include the following:

- We will see an example (string equality) in which randomised algorithms have provably better *communication complexity* than deterministic algorithms.
- Randomised algorithms sometimes provide *provably better guarantees* in online algorithms and learning.
- Randomised algorithms sometimes provide *provably better guarantees* in approximation algorithms. (A famous example, which is beyond the scope of this class, is approximating the volume of a convex body.)

We will see in Section 1.2.2 a major open problem in computational complexity which is about determining whether or not randomised algorithms are better than deterministic ones in the context of polynomial-time decision algorithms.

### 1.1.2 String equality

**String equality problem** Alice has a binary string $a$ and Bob has a binary string $b$, both of length $n$. They want to check whether their strings are the same with minimal communication between them. (This makes sense when $n$ is large, say in the billions.)

The straightforward method is for Alice to transmit her string to Bob who performs the check and informs Alice. This requires a communication of $\Theta(n)$ bits. There is a better way to do it, which is based on the following reduction.

- Alice and Bob create two polynomials out of their strings: $f(x) = \sum_{i=0}^{n-1} a_i x^i$ and $g(x) = \sum_{i=0}^{n-1} b_i x^i$.
- In doing so, they reduce the string equality problem to the problem of checking equivalence of univariate polynomials of degree $n-1$, because the polynomials are the same if and only if they have the same coefficients.

**Checking equivalence of polynomials** Given two (univariate) polynomial $f(x)$ and $g(x)$ of degree $n$, check whether they are the same.

- More generally, we might not assume that the polynomials are in expanded form. For example, we might have $f(x) = (x-3)(x^3 - 2x + 1)^3 - 7$.

**Idea for algorithm** Select a large random integer $r$ and check whether $f(r) = g(r)$.

**Immediate question** How large does $r$ need to be in order to ensure that the probability of answering incorrectly is at most $\varepsilon$?

- clearly there is a tradeoff between the range of $r$ and the probability of error
- note the asymmetry (*one-sided error*): if the polynomials are the same, the algorithm is *always* correct
- if the polynomials are not the same, the algorithm returns the wrong answer exactly when $r$ is a root of the polynomial $f(x) - g(x)$
- a univariate polynomial of degree at most $n$ that is not identically 0 has at most $n$ roots. If we select $r$ to be a random integer in $\{1, 100n\}$, the probability of error is at most $1/100$.

**Running time?** Note that the value of the polynomial, $f(r)$, can be as high as $(100n)^{n-1}$. Alice has to send this to Bob. The intermediate results also may need $\Omega(n \log n)$ bits.

- Would it not be better to transmit the original string instead of $f(r)$?
- The answer is no, because there is a clever way to reduce the size of numbers: do all computations in a finite field $F_p$, for some prime $p \geq 100n$.

**Algorithm** Here is an algorithm that implements this idea:

*Select a prime $p$ larger than $100n$ and a random integer $r \in \{0, \ldots, p-1\}$. Check whether*

$$f(r) = g(r) \pmod{p}.$$

Using this algorithm, Alice and Bob can solve the string equality problem as follows: Alice sends to Bob $(r, f(r) \pmod{p})$. Bob checks whether $f(r) = g(r) \pmod{p}$ and announces the result.

**Complication?** We don't want to let $p$ be too big, because Alice still has to send $p$ to Bob. Fortunately, we can be sure that there is a prime number between $100n$ and $200n$. In fact, there are lots of them.[1]

**Refined Algorithm** Here is a better version of the algorithm:

*Select a prime $p$ between $100n$ and $200n$ and a random integer $r \in \{0, \ldots, p-1\}$. Check whether*

$$f(r) = g(r) \pmod{p}.$$

Again, Alice and Bob can solve the string equality problem as follows: Alice sends to Bob $(r, f(r) \pmod{p})$. Bob checks whether $f(r) = g(r) \pmod{p}$ and announces the result.

- Now only $O(\log n)$ bits are sent.
- The time that it takes to evaluate the polynomials is at most a polynomial in $n$.[2]
- the argument for correctness is similar to the one that we already gave; we only replaced the field of reals with $F_p$:
  - First, notice that if the polynomials are equal, the algorithm is always correct.
  - Otherwise, $f(x) - g(x)$ is a nonzero polynomial of degree at most $n$.
  - But a polynomial of degree $n < p$ has at most $n$ roots in $F_p$. Therefore the probability of an incorrect result is at most $n/p \leq 1/100$.
  - Why do we need $p$ to be prime? Because if $p$ is not prime, a polynomial of degree $n$ may have more than $n$ roots. For example, the second-degree polynomial $x^2 - 1$ has four roots $\mod 8$ (any odd number will do).

**Loose ends**

- Can Alice find a prime number efficiently? Yes, by repeatedly selecting a random integer between $100n$ and $200n$ until a prime number is found
  - We can check whether a number $q$ is prime in time polynomial in $\log q$. The simplest known algorithms for this are randomised algorithms.
- Is probability $1/100$ sufficient?
  - We can increase the parameter $100n$ to get a better probability; if we replace it with $1000n$ the error probability will drop to $1/1000$.
  - There is a much better way: run the test $k$ times, with independent numbers $r$. This will make the probability of error $1/100^k$ (why?)

[1] This follows from the Prime number theorem. For a real number $x$, let $P(x)$ be the number of primes that are at most $x$. Let $Q(x) = x/\log(x)$. The theorem shows that $\lim_{x\to\infty} P(x)/Q(x) = 1$. Note that $Q(2x) - Q(x) = x(2\log(x) - \log(2x))/(\log(2x)\log(x))$.

[2] In fact, we can write $f(x) = \sum_{i=0}^{n-1} a_i x^i$ as $a_0 + x(a_1 + x(a_2 + x(a_3 + \cdots + x(a_{n-2} + xa_{n-1}) \cdots )))$ to evaluate the polynomial in $O(n)$ operations. This is called "Horner's method."

  - In retrospect, and with this *amplification of probability* in mind, it is better to run the original test with a prime number $p$ between $2n$ and $4n$ to achieve one-sided probability of error at most $1/2$, and repeat as many times as necessary to achieve the desired bound on error

- We have seen that Alice and Bob can verify the equality of two $n$-bit strings by communicating $O(\log n)$ bits. Every deterministic algorithm requires $n$ bits (why?).

### Recap of useful ideas

**fingerprinting** verify properties by appropriate random sampling

- useful trick: perform calculations modulo a prime number to keep the numbers small

**probability amplification** run many independent tests to decrease the probability of failure

- for one-sided error, we usually design algorithms with probability of error $1/2$. Using probability amplification, we can easily decrease it to any desired probability bound

### Further topics

- it is not hard to extend the algorithm for checking polynomial identities to polynomials of multiple variables. The only difficulty is to argue about the number of roots of multi-variate polynomials (check out the Schwartz-Zippel lemma).

- we can extend string equality to string matching (i.e. given two strings $a$ and $b$ check whether $b$ is a substring of $a$), a very useful application. The obvious way is to check whether $b$ appears in the first, second and so on position of $a$. The clever trick is to find a way to combine the evaluation of the associated polynomials in an efficient way (check out the Rabin-Karp algorithm).

- applications that use efficient checking of polynomials come from unexpected domains. The perfect matching problem is such an example. The problem asks whether a given graph has a perfect matching, i.e., a set of disjoint edges that cover all vertices. The question is equivalent to whether the determinant of the Tutte matrix of the graph is not the zero polynomial (check out Tutte matrix).

### Sources

- Mitzenmacher and Upfal, 2nd edition, Section 1.1
- Lap Chi Lau, L01.pdf (page 6)
- Nick Harvey, Lecture1Notes.pdf (page 3)
- James Lee, lecture1.pdf (page 1)

## 1.2 Minimum-size cut-set contraction algorithm

We will see now a strikingly simple randomised algorithm for a non-trivial graph problem.

### 1.2.1 Minimum-size cut-set contraction algorithm

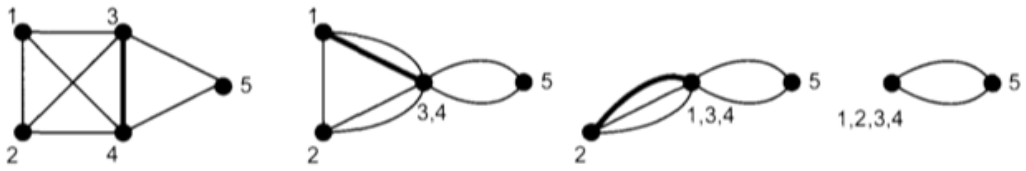
**The min-size cut-set problem** Given a graph $G = (V, E)$, find a partition of the vertices into two non-empty parts in a way that minimises the number of edges from one part to the other.

- A cut-set of a graph $G = (V, E)$ is a set $A \subseteq E$ such that the graph $(V, E \setminus A)$ has at least two connected components. (Notation: $G \setminus A$ means $(V, E \setminus A)$.)
- In the *min-size cut-set problem* the input is a graph $G$ and the output is a cut-set $A$ of $G$ whose size $|A|$ is as small as possible.
- If $G$ has more than one connected component then the solution is $A = \emptyset$.
- If $G$ is connected then, for any minimum-size cut-set $A$ of $G$, $G \setminus A$ has exactly two connected components.

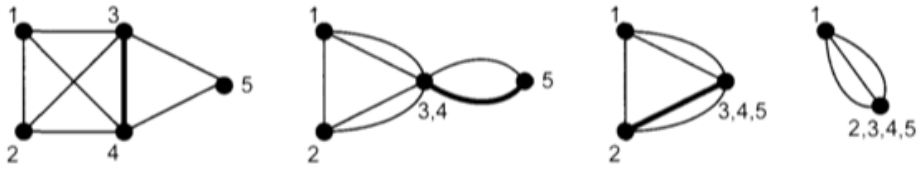
**Other algorithms** If you took our Algorithms course then you learned about the Ford-Fulkerson algorithm. This algorithm takes as input the graph $G = (V, E)$, but also two distinct vertices $s$ and $t$. The output is a cut-set $A$ of $G$ such that $s$ and $t$ are in difference components in $G \setminus A$. Once again, $A$ is chosen to minimise $|A|$. If $n = |V|$ and $m = |E|$ then the algorithm takes $O(nm)$ time. Given a connected graph $G$, we could solve the min-size cut-set problem by solving $\binom{n}{2}$ instances of the $s$-$t$ min-size cut-set problem and taking the best. Actually, it suffices to solve $n - 1$ instances of the $s$-$t$ min-size cut-set problem. Take any vertex $s$ and there will be some $t$ in the other component. So this takes $O(n^4)$ time.

**Karger's idea** Let $G$ be a connected graph with at least two vertices in which we desire a minimum-size cut-set. Form $G'$ from $G$ by picking a random edge of $G$ and contracting it (throw away the new self-loop between its end-points, but keep any parallel edges that the contraction creates). If $A$ is a cut-set of $G'$ then it is a cut-set of $G$. What about the other direction? If $A$ is a cut-set of $G$, then it will be a cut-set of $G'$ *unless the contracted edge was in* $A$.

In order to (hopefully) find a minimum-size cut-set of $G$, we'll keep contracting random edges, until we get down to two vertices, and then take the cut-set between them. Consider some minimum-size cut-set $A$ of $G$. As long as edges in $A$ are never contracted, it will be the output.




(a) A successful run of min-cut.




(b) An unsuccessful run of min-cut.

**Correctness** Let $G$ be a connected multi-graph (it may have parallel edges, but no self-loops). Suppose that it has $m$ edges, and fix a minimum-size cut-set $A$. What is the probability that $A$ will survive the first edge contraction?

- Let $k = |A|$.
- Then $m \geq kn/2$. (Why?)[3]
- The probability that an edge of $A$ is selected is $k/m$, which is at most $2/n$.
- The probability that $A$ survives the first contraction is at least $1 - 2/n = (n-2)/n$.

What is the probability that $A$ will survive all contractions until only two vertices remain?

$$\frac{n-2}{n}\frac{n-3}{n-1}\frac{n-4}{n-2}\cdots\frac{2}{4}\frac{1}{3} = \frac{2}{n(n-1)}$$

- thus the probability that the contraction algorithm will be incorrect is at most the probability that it doesn't output $A$, which is at most $1 - \frac{2}{n(n-1)} \leq 1 - \frac{2}{n^2}$
- this may seem very close to 1, but if we run the contraction algorithm $n^2$ times and keep the best result, the probability that it will not return a minimum-size cut-set (in fact, the particular minimum-size cut-set $A$) will drop to [4]

$$\left(1 - \frac{2}{n^2}\right)^{n^2} \leq \left(e^{-\frac{2}{n^2}}\right)^{n^2} = e^{-2} < \frac{1}{2}$$

Therefore we have:

**Theorem 1.2.1.** *If we repeat the contraction algorithm $n^2$ times, the probability that it will find a minimum-size cut-set is at least $1/2$.*

[3] Each vertex $v$ has degree at least $k$ since otherwise the set of edges incident to $v$ would be a cut-set of size less than $k$. But then $2m = \sum_v \deg(v) \geq nk$.

[4] We use the fact that $e^x \geq 1 + x$, for every $x$. Proof: Let $f(x) = e^x - 1 - x$. Then $f'(x) = e^x - 1$, which is $0$ at $x = 0$. $f''(x) = e^x > 0$ so $f(x)$ is minimised at $x = 0$. Thus, $f(x) \geq f(0) = 0$.

**Running time** Suppose that we store

- an $n \times n$ matrix $W$ where $W(u, v)$ is the number of edges between $u$ and $v$
- a length-$n$ array $D$ where $D(u)$ is the degree of $u$.

Then one contraction takes $O(n)$ times as follows.

- Choosing an edge to contract
  - Choose $u$ with probability proportional to $D(u)$ in $O(n)$ time.
  - Choose $v$ with probability proportional to $W(u, v)$ in $O(n)$ time.
- Contracting an edge: Update the data structures in $O(n)$ time.

So the contraction algorithm takes $O(n^2)$ time and running it $n^2$ times takes $O(n^4)$ time in all, the same as the algorithm that we mentioned earlier, based on computing $s$-$t$ min-size cut-sets.

**Improving the running time**

- the probability that the cut-set $A$ survives decreases as the graph becomes smaller
- to take advantage of this, we stop the contractions earlier – say when the graph has size $r = r(n)$ – and switch to another min-size cut-set algorithm
- the probability of success becomes $r(r-1)/n(n-1)$ and we need to repeat it only $O(n^2/r^2)$ times to achieve probability of success at least $1/2$
- on the other hand, we must take into account the execution time of the algorithm we switch to when the graph reaches size $r$
- which algorithm should we switch to? One possibility is the same algorithm (recursion) but with repeats, to boost the success probability.

**Algorithm** Here is a variant of the above idea due to Karger and Stein:

*Recursive-Contract(G)*

- *if* $n \leq 6$ *use brute-force*
- *else repeat* ***twice*** *and return the best result*
  - *G' = Contract(G,* $\lfloor n/\sqrt{2} \rfloor$*)*
  - *Recursive-Contract(G')*

*Contract(G, r)*

- *repeat until the graph has* $\leq r$ *vertices*
  - *select a random edge and contract it*

They show that

- Algorithm Recursive-Contract runs in $O(n^2 \log n)$ time.
- The probability that Recursive-Contract succeeds is $\Omega(1/\log(n))$

If the success probability is at least $1/\log(n)$ then, running it $\log(n)$ times, the overall failure probability is at most $(1 - 1/\log n)^{\log n} \leq 1/e$. The overall running time is then $O(n^2(\log n)^2)$. We won't cover the details in this course, but if you are interested, see the original paper "A new approach to the minimum cut problem".

**Number of minimum-size cut-sets** As an extra bonus of the analysis of the contraction algorithm, we get that every graph has at most $n(n-1)/2$ minimum-size cut-sets. Why? Show also that this is tight by finding a graph that has exactly $n(n-1)/2$ minimum-size cut-sets (you can do this for any $n$).

**Better algorithms** There are better randomised and deterministic min-size cut-set algorithms than the one presented here, but the one presented here is nice and simple.

**Sources**

- Mitzenmacher and Upfal, 2nd edition, Section 1.5

### 1.2.2 What is a randomised algorithm?

A randomised algorithm can be thought of as a deterministic algorithm $A(x, r)$ with two inputs. $x$ is the "real" input and $r$ is a sequence of random bits. Informally, we say that $A(x, r)$ computes a function $f(x)$ if $A(x, r)$ output $f(x)$ with significant probability, when $r$ is selected uniformly at random (u.a.r.)[5] amongst bit-strings of some given length (depending on $x$).

Recall that a language is a set of strings over some fixed alphabet.

**Definition 1.2.2.** A language $L$ is in Randomised-Polynomial-time (RP) if there is a polynomial-time deterministic algorithm $A(x, r)$ whose bit sequence $r$ has length $p(|x|)$, for some polynomial $p(\cdot)$, and for every $x$:

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

We view this as a randomised algorithm because, when we provide the algorithm with a random $r$, it will accept with probability at least $1/2$, when $x \in L$, and probability $0$, when $x \notin L$.

- Note the asymmetry between positive and negative instances. When an RP algorithm accepts, we are certain that $x \in L$; when it rejects, we don't know with certainty.
- The class co-RP is the class with the same definition as RP, but with the roles of "accept" and "reject" inverted. For example, the string equality problem is in co-RP, and the string inequality problem in RP.

[5] This is an abbreviation that is used often in these notes. Remember that u.a.r. means "uniformly at random".

- The bound $1/2$ on the probability is arbitrary; any constant in $(0, 1)$ defines the same class.
- The class NP can actually be defined in a similar way. The only difference is that, when $x \in L$, the algorithm accepts not for at least half of the $r$'s, but for at least a single $r$.
- Similarly, we could define the class P in a similar way. The only difference is that, when $x \in L$, the algorithm accepts for every $r$.

This shows

**Theorem 1.2.3.**

$$\mathrm{P} \subseteq \mathrm{RP} \subseteq \mathrm{NP}.$$

**Powering** For RP algorithms, if we want higher probability of success, we run the algorithm $k$ times, so that the algorithm succeeds for $x \in L$ with probability at least $1 - 2^{-k}$.

**Open problem** It is an open problem whether RP is equal to P. Similarly, it is open whether RP is equal to NP.

**Other Randomised complexity classes** The class ZPP (Zero-error Probabilistic Polynomial time) is the intersection of RP and its complement co-RP. Note that a language in ZPP has an RP and a co-RP algorithm. We can repeatedly run both of them until we get an answer for which we are certain. The classes RP and ZPP correspond to the following types of algorithms

**Monte Carlo algorithm (RP)** it runs in polynomial time and has probability of success at least $1/2$ when $x \in L$, and its is always correct when $x \notin L$.

**Las Vegas algorithm (ZPP)** it runs in *expected* polynomial time and it always gives the correct answer.

Arguably, if we leave aside quantum computation, the class that best captures feasible computation is the BPP class:

**Definition 1.2.4.** A language $L$ is in Bounded-error Probabilistic-Polynomial-time (BPP) if there is a polynomial-time deterministic algorithm $A(x, r)$ whose bit sequence $r$ has length $p(|x|)$, for some polynomial $p(\cdot)$, and for every $x$:

- if $x \in L$, $A(x, r)$ accepts for at least $3/4$ of the $r$'s
- if $x \notin L$, $A(x, r)$ rejects for at least $3/4$ of the $r$'s.

- Again the constant $3/4$ is arbitrary; any constant in $(1/2, 1)$ defines the same class.
- By repeating the algorithm many times and taking the majority, we can bring the probability of getting the correct answer very close to 1.

We don't know much about the relation between the various classes. Clearly RP $\subseteq$ BPP, because if we run an RP algorithm twice, the probability of success is at least $3/4$, for every input. One major open problem is whether BPP $\subseteq$ NP. It is unlikely that NP $\subseteq$ BPP since this would cause the whole polynomial hierarchy to collapse down to BPP. (That is beyond the scope of this course, though!) Many conjecture that P = BPP.[6]

**Sources**

- Luca Trevisan's notes lecture08.pdf

[6] One reason for this conjecture is the result of Impagliazzo and Wigderson at https://doi.org/10.1145/258533.258590. The worst-case circuit complexity of a problem is a function $f$ such that $f(n)$ is the minimum number of gates in a circuit that computes the answer on inputs of size $n$. Their result shows that if any problem in $\mathrm{DTime}(2^{O(n)})$ has worst-case circuit complexity $2^{\Omega(n)}$ then P = BPP. That paper also gives discusses some other assumptions that are known to imply P = BPP.

PART 2
LINEARITY OF EXPECTATIONS

## 2.1 Linearity of expectations

**Theorem 2.1.1.** *Suppose that $X$ and $Y$ are discrete random variables with finite expectations. Then*

$$\mathbb{E}[X+Y] = \mathbb{E}[X] + \mathbb{E}[Y].$$

*In general, if $X_1, \ldots, X_n$ are random variables with finite expectations, then*

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

*Remark.* Note that this theorem does not require the variables $X$ and $Y$ to be independent!

*Remark.* Most of the random variables that we will encounter in this course are discrete, so probabilistic calculations are equivalent to counting arguments. Even in the discrete case, the probabilistic perspective gives an intuitive and algorithmically efficient way to reach conclusions. One of the themes of this course is that very simple arguments, such as independence and linearity of expectations, when used appropriately, can lead to very powerful algorithms. In this section, we'll see two nice examples of uses of linearity of expectations.

### 2.1.1 Max Cut

**The Max-Cut problem**

- A cut of a graph $G = (V, E)$ is represented by a *subset $C$ of $V$*. Its *value* is the *number of edges* from $C$ to $V \setminus C$.
- A *max cut* is a cut whose value is as large as possible.

Finding a max cut is known to be an NP-hard problem. It is also an APX-hard problem, meaning that unless P=NP, there is no approximation algorithm with approximate ratio arbitrarily close to 1.

*With respect to approximability, a maximisation problem may*

- *have a polynomial time algorithm*
- *have a PTAS (polynomial-time approximation scheme): for every $\varepsilon > 0$, there is a polynomial time algorithm that finds a solution within a factor $1-\varepsilon$ of the optimum. For example, a PTAS can have running time $O(n^{1/\varepsilon})$*

  - *have a FPTAS (fully polynomial-time approximation scheme): a PTAS whose running time is also polynomial in $1/\varepsilon$, for example $O(n/\varepsilon^2)$*

- *be APX-hard, i.e. it has no PTAS (assuming $\mathrm{P} \neq \mathrm{NP}$). Assuming $\mathrm{P} \neq \mathrm{NP}$, APX-hardness does not mean that a problem cannot be approximated at all, but rather that it cannot be approximated to any desired degree. For example, we will now see that Max Cut can be approximated within a factor of $1/2$. It is known that Max Cut is APX-hard and that (unless $\mathrm{P} = \mathrm{NP}$) there is no polynomial-time algorithm to find solutions within a factor better than $16/17$ of the optimal solution.*

For a *minimisation* problem the definitions are similar except that a PTAS should find a solution that is within a $1 + \varepsilon$ factor of being optimal.

Together, maximisation problems and minimisation problems are called *optimisation problems*.

**Approximate algorithm for Max Cut** There are many simple polynomial-time algorithms with approximation ratio $1/2$. Here we will discuss a randomised algorithm:

*Select a random cut, i.e., every vertex is put in $C$ with probability $1/2$.*

This algorithm is so simple that it produces a cut without even looking at the graph!

### Analysis

**Theorem 2.1.2.** *For every graph with $m$ edges, the expected value of the cut produced by the algorithm is $m/2$.*

*Proof.* Let $C$ be the cut produced by the algorithm. We will show that the expected number of edges from $C$ to $V \setminus C$ is $m/2$, where $m$ is the number of edges of the graph.

Fix an edge $(u, v)$. Let $X_{u,v}$ be an *indicator random variable* that takes value 1 if $(u, v)$ is an edge from $C$ to $V \setminus C$ and 0 otherwise.

The probability that $X_{u,v} = 1$ is

$$\mathbb{P}((u \in C \wedge v \notin C) \vee (u \notin C \wedge v \in C)) = \frac{1}{2} \cdot \frac{1}{2} + \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{2}.$$

Therefore $\mathbb{E}[X_{u,v}] = \frac{1}{2} \cdot 1 + \frac{1}{2} \cdot 0 = \frac{1}{2}$.

Let $X = \sum_{(u,v) \in E} X_{u,v}$ be the value of cut $C$. Then $E[X]$ is

$$\begin{aligned} \mathbb{E}\left[\sum_{(u,v) \in E} X_{u,v}\right] &= \sum_{(u,v) \in E} \mathbb{E}[X_{u,v}] \\ &= \sum_{(u,v) \in E} \frac{1}{2} \\ &= \frac{m}{2} \end{aligned}$$

Notice the use of the linearity of expectations. □

Since the optimum cut cannot have more than $m$ edges, the approximation ratio of our algorithm is at least $1/2$.

The above analysis shows that the expected cut is of value $m/2$, but it provides no information about the probability distribution of the value of the cut. An interesting question is how many times, in expectation, we need to repeat the algorithm in order to be likely to get a cut of value at least $m/2$.

Here is a simple way to compute such a bound. Let $p$ be the probability that we obtain a cut of value at least $m/2$ when we run the algorithm. Let's first assume that $m$ is even (so $m/2$ is an integer). Then we have

$$\begin{aligned}\frac{m}{2} = \mathbb{E}[X] &= \sum_{i<m/2} i \cdot \mathbb{P}(X=i) + \sum_{i\geq m/2} i \cdot \mathbb{P}(X=i) \\ &\leq \left(\frac{m}{2}-1\right)\mathbb{P}\left(X \leq \frac{m}{2}-1\right) + m\,\mathbb{P}\left(X \geq \frac{m}{2}\right) \\ &= \left(\frac{m}{2}-1\right)(1-p) + mp \\ &= \left(\frac{m}{2}-1\right) + \left(\frac{m}{2}+1\right)p\end{aligned}$$

We conclude that $p \geq \frac{1}{m/2+1}$, and hence that the expected number of trials before finding a cut of value of at least $m/2$ is at most $m/2+1$. (We will consider the *geometric distribution* in Section 2.3.1, but for now it is enough to note that if each trial succeeds with probability $q = 1/(m/2+1)$ then the expected number of trials until success is $1/q = m/2+1$.)

The case where $m$ is odd is similar.

**The probabilistic method** The analysis shows that in every graph there exists a cut which is crossed by at least half of the the edges of the graph.

This statement is a typical result of the *probabilistic method* in which we show existence of an object with a certain property by showing that with respect to some distribution over the objects, either

- a random object has the property with non-zero probability, or
- the expected number of objects with the property is positive

### 2.1.2 MAX-3-SAT

**The MAX-SAT problem** Given a Boolean CNF formula[1], find a truth assignment that maximises the number of satisfied clauses.

This is an extension of the Boolean CNF satisfiability decision problem and it is known to be NP-hard. It is also APX-hard, i.e., there is no PTAS for this problem unless P=NP.

There are interesting special versions of the problem: MAX-2-SAT, MAX-3-SAT, and in general MAX-k-SAT in which the input is a k-CNF formula, where each clause has exactly $k$ literals (whose variables are distinct). All of these special cases are NP-hard and APX-hard.

**Randomised algorithm for MAX-3-SAT** The following is a simple algorithm for the MAX-SAT problem.

[1]The formula is an AND of clauses. Each clause is the OR of literals. Each literal is either a variable, or its negation.

*Select a random assignment, i.e., set every variable independently to true with probability* $1/2$.

*Remark.* Again, this is so simple that one can hardly consider it an algorithm; it decides the output without looking at the clauses. Not all randomised algorithms are so simple, but in general they are simpler than their deterministic counterparts.

**Analysis** We will analyse the performance of the algorithm for the special case of MAX-3-SAT. A similar analysis applies to the general MAX-SAT problem, and we will revisit it later when we discuss how to "derandomise" this algorithm.

**Theorem 2.1.3.** *For every 3-CNF formula with* $m$ *clauses, the expected number of clauses satisfied by the above algorithm is* $\frac{7}{8}m$.

*Proof.* Let $\varphi = C_1 \wedge \cdots \wedge C_m$ be a formula with clauses $C_1, \ldots, C_m$. Let $X_i$, $i = 1, \ldots, m$, be an indicator random variable, indicating whether clause $C_i$ is satisfied or not in a (random) assignment. The probability that $X_i$ is $1$ is equal to $7/8$, because out of the $8$ possible (and equiprobable) assignments to the three distinct variables of $C_i$ only one does not satisfy the clause, i.e. $\mathbb{P}(X_i = 1) = 7/8$.

From this we get $\mathbb{E}[X_i] = 7/8 \cdot 1 + 1/8 \cdot 0 = 7/8$ and by linearity of expectations:

$$\begin{aligned}\mathbb{E}\left[\sum_{i=1}^{m} X_i\right] &= \sum_{i=1}^{m} \mathbb{E}[X_i] \\ &= \sum_{i=1}^{m} \frac{7}{8} \\ &= \frac{7}{8}m\end{aligned}$$

□

Note the similarity between this proof and the proof for approximating Max Cut. Interpreting this result with the probabilistic method, we see that for every 3-CNF formula there is a truth assignment that satisfies at least $7/8$ of its clauses.

**Sources**

- Mitzenmacher and Upfal, 2nd edition, Section 6.2

## 2.2 Derandomisation

### 2.2.1 Derandomisation (method of conditional expectations)

Derandomisation is the process of taking a randomised algorithm and transforming it into an equivalent deterministic one without increasing substantially its complexity. Is there a generic way to do this? We don't know. In fact, we have randomised algorithms that we don't believe can be transformed directly into deterministic ones (for example, the algorithm of verifying identity of multivariate polynomials). But we do have some techniques that work

for many randomised algorithms. The simplest such technique is the *method of conditional expectations*. Other more sophisticated methods of derandomisation are based on reducing the number of required random bits, and then trying all possible choices for these random bits.

### Derandomisation using conditional expectations

To illustrate the idea of the method of conditional expectations, suppose that we have a randomised algorithm that randomly selects values for random variables $X_1, \ldots, X_n$ with the aim of maximising $\mathbb{E}[f(X_1, \ldots, X_n)]$. The method of conditional expectations fixes one-by-one the values of the random variables $X_1, \ldots, X_n$ to be $x_1^*, \ldots, x_n^*$ so that $f(x_1^*, \ldots, x_n^*) \geq \mathbb{E}[f(X_1, \ldots, X_n)]$. To do this, we compute

$$\begin{aligned}
x_n^* &= \operatorname*{argmax}_x \mathbb{E}[f(X_1, \ldots, X_n) \mid X_n = x] \\
x_{n-1}^* &= \operatorname*{argmax}_x \mathbb{E}[f(X_1, \ldots, X_n) \mid X_n = x_n^*, X_{n-1} = x] \\
x_{n-2}^* &= \operatorname*{argmax}_x \mathbb{E}[f(X_1, \ldots, X_n) \mid X_n = x_n^*, X_{n-1} = x_{n-1}^*, X_{n-2} = x] \\
&\vdots \\
x_1^* &= \operatorname*{argmax}_x \mathbb{E}[f(X_1, \ldots, X_n) \mid X_n = x_n^*, \ldots, X_2 = x_2^*, X_1 = x]
\end{aligned}$$

to obtain successively values $x_n^*, \ldots, x_1^*$ for which $f$ takes a value at least as big as its expectation. If we can compute the above deterministically and efficiently, we obtain a deterministic algorithm that is as good as the randomised algorithm.

In many cases, the random variables $X_i$ are indicator variables, which makes it easy to compute the argmax expression by computing the conditional expectation given $X_i = 0$ and $X_i = 1$ and selecting the maximum.

### Derandomisation of the MAX-SAT algorithm

Recall the MAX-SAT algorithm which simply selects a random truth assignment. We now show how to derandomise this algorithm. When we analysed the approximation ratio for MAX-SAT, we did it only for MAX-3-SAT because it was simpler. But for the discussion here, it is easier to consider general CNF formulas. In general, the inductive approach for derandomising a problem with the method of conditional expectations is facilitated by an appropriate generalisation of the problem at hand.

The crucial observation is:

**Lemma 2.2.1.** *For every CNF formula with clauses of sizes $c_1, \ldots, c_m$, the expected number of clauses satisfied by a random truth assignment is $\sum_{i=1}^{m}(1 - 2^{-c_i})$.*

*Proof.* The probability that the $i$ 'th clause is satisfied is $1 - 2^{-c_i}$, because out of the $2^{c_i}$ truth assignments to the variables of the clause[2], only one does not satisfy the clause. If $Z_i$ is an indicator random variable that captures whether clause $i$ is satisfied, then $\mathbb{E}[Z_i] = 1 - 2^{-c_i}$ (we used a similar reasoning when we analysed the MAX-3-SAT algorithm). We want to compute $\mathbb{E}[\sum_{i=1}^{m} Z_i]$. With the linearity of expectations this is $\sum_{i=1}^{m} \mathbb{E}[Z_i] = \sum_{i=1}^{m}(1 - 2^{-c_i})$. □

[2] We assume that all the variables of a clause are distinct.

Given this, we can design an algorithm that satisfies at least so many clauses as in the lemma. It is best illustrated by an example.

Let $\varphi(x_1, x_2, x_3) = (x_1 \vee \overline{x}_2) \wedge (\overline{x}_1 \vee \overline{x}_3) \wedge (x_1 \vee x_2 \vee x_3)$. A random truth assignment satisfies $1 - 2^{-2} + 1 - 2^{-2} + 1 - 2^{-3} = 19/8$ clauses, in expectation.

If we fix the value of $x_3$ to

**true** the resulting formula is $(x_1 \vee \overline{x}_2) \wedge (\overline{x}_1) \wedge \text{true}$. A random assignment satisfies $1 - 2^{-2} + 1 - 2^{-1} + 1 = 9/4$ clauses.

**false** the resulting formula is $(x_1 \vee \overline{x}_2) \wedge \text{true} \wedge (x_1 \vee x_2)$. A random assignment satisfies $1 - 2^{-2} + 1 + 1 - 2^{-2} = 5/2$ clauses.

We must have $\frac{1}{2}\frac{9}{4} + \frac{1}{2}\frac{5}{2} = \frac{19}{8}$, which implies that the maximum of the values $9/4$ and $5/2$ (the two conditional expectations) must be at least $19/8$. Indeed, $5/2 \geq 19/8$. This suggests that we set $x_3$ to false and repeat with the resulting formula $(x_1 \vee \overline{x}_2) \wedge \text{true} \wedge (x_1 \vee x_2)$.

We can generalise this approach to every CNF formula and we can turn it into the following algorithm:

*max-sat*$(\varphi(x_1, \ldots, x_n))$

- *Let* $c_1^1, \ldots, c_k^1$ *be the sizes (not counting* $x_n$*) of all clauses containing literal* $x_n$
- *Let* $c_1^0, \ldots, c_l^0$ *be the sizes (not counting* $\overline{x}_n$*) of all clauses containing literal* $\overline{x}_n$
- *If* $\sum_{i=1}^{k} 2^{-c_i^1} \geq \sum_{i=1}^{l} 2^{-c_i^0}$ *then set* $x_n =$ *true, else set* $x_n =$ *false*
- *Repeat for the formula* $\varphi'(x_1, \ldots, x_{n-1})$ *that results when we fix* $x_n$

Note that we simplified the comparison of the expectations for the two cases $x_n = \text{true}$ and $x_n = \text{false}$ to $\sum_{i=1}^{k} 2^{-c_i^1} \geq \sum_{i=1}^{l} 2^{-c_i^0}$.

**Theorem 2.2.2.** *For every 3CNF formula, the above polynomial-time deterministic algorithm returns a truth assignment that satisfies at least* $7/8$ *of the clauses.*

It is known that unless P=NP, this is the best approximation ratio achievable by any polynomial-time algorithm.

### Derandomisation of the Max-Cut algorithm

Recall the randomised algorithm for Max Cut which partitions the vertices of the graph randomly. To derandomise it, we will fix these random choices one-by-one. To do this, let $X_i$ be an indicator random variable, indicating whether vertex $i$ is in the cut $C$, and let $c(X_1, \ldots, X_n)$ be a random variable that gives the value of the cut. The derandomisation is based on the computation of $\mathbb{E}[c(X_1, \ldots, X_n) \mid X_i = x_i, \ldots, X_n = x_n]$, which is the expected value of the cut after we fix the position of vertices $i, \ldots, n$. An edge contributes to this expectation:

- 1, if both vertices have been fixed and are in different parts
- 0, if both vertices have been fixed and are in the same part
- $1/2$, if at least one of the vertices has not been fixed,

from which we get the following claim.

**Claim 2.2.3.**

$$\mathbb{E}[c(X_1,\ldots,X_n)|X_i=x_i,\ldots,X_n=x_n] = \sum_{(u,v)\in E} \begin{cases} 1 & \text{if } u,v \geq i \text{ and } x_u \neq x_v \\ 0 & \text{if } u,v \geq i \text{ and } x_u = x_v \\ \frac{1}{2} & \text{otherwise.} \end{cases}$$

The algorithm then is:

*Max-Cut(G)*

- *For* $i = n,\ldots,1$
  - *if* $\mathbb{E}[c(X_1,\ldots,X_n)|X_i=1,X_{i+1}=x_{i+1},\ldots,X_n=x_n]$ *is greater than or equal to* $\mathbb{E}[c(X_1,\ldots,X_n)|X_i=0,X_{i+1}=x_{i+1},\ldots,X_n=x_n]$ *then fix* $x_i=1$ *and put vertex* $i$ *into the cut* $C$.
  - *Otherwise, fix* $x_i=0$ *and do not put vertex* $i$ *into* $C$.

A closer look at this algorithm shows that it is essentially the greedy algorithm: it processes vertices $n,\ldots,1$ and for each one of them, it puts it in the part that maximises the value of the maximum-value cut in the graph induced by the already-processed vertices.

**Theorem 2.2.4.** *For every graph* $G$*, the above polynomial-time deterministic algorithm returns a cut of value at least half the total number of edges.*

**Sources**

- Mitzenmacher and Upfal, 2nd edition, Sections 6.3

### 2.2.2 Pairwise independence and derandomization

Recall that random variables $X_1,\ldots,X_n$ are *independent* if, for all $a_1,\ldots,a_n$,

$$\mathbb{P}(X_1=a_1\wedge\cdots\wedge X_n=a_n) = \mathbb{P}(X_1=a_1)\cdots\mathbb{P}(X_n=a_n).$$

**Definition 2.2.5.** We say that $X_1,\ldots,X_n$ are *pairwise independent* if, for every pair of distinct variables $X_i$ and $X_j$ in $\{X_1,\ldots,X_n\}$,

$$\mathbb{P}(X_i=a_i\wedge X_j=a_j) = \mathbb{P}(X_i=a_i)\cdot\mathbb{P}(X_j=a_j).$$

**Bits for free!**

A random bit is uniform if it assumes the values 0 and 1 with equal probability. Suppose that $X_1,\ldots,X_n$ are independent uniform random bits. Given a non-empty set $S\subseteq\{1,\ldots,n\}$,

define $Y_S = \bigoplus_{i \in S} X_i$ (where $\oplus$ denotes *exclusive or*). Clearly $\mathbb{P}(Y_S = 1) = 1/2$. [3] Similarly, given two different non-empty sets $S, T \subseteq \{1, \ldots, n\}$, we have for every $i, j \in \{0, 1\}$:

$$\mathbb{P}(Y_S = i \wedge Y_T = j) = 1/4.$$

Therefore

**Claim 2.2.6.** The $2^n - 1$ random variables $Y_S = \bigoplus_{i \in S} X_i$, where $S$ a non-empty subset of $\{1, \ldots, n\}$ are pairwise independent.

Note though that the $Y_S$'s are not fully independent since, e.g., $Y_{\{1\}}$ and $Y_{\{2\}}$ determine $Y_{\{1,2\}}$.

#### Application: Finding max cuts

Let $G = (V, E)$ be an undirected graph with $n$ vertices and $m$ edges. Recall the argument that $G$ has a cut of value at least $m/2$. Suppose that we choose $C \subseteq V$ u.a.r. Let $X$ be the number of edges with one endpoint in $C$ and the other in $V \setminus C$. Then $\mathbb{E}[X] = m/2$ by linearity of expectations. We conclude that there must be some cut of value at least $m/2$.

The above existence result is not constructive—it does not tell us how to find the cut. We have seen how to derandomise this algorithm using conditional expectations. Here we see another way to derandomise the algorithm using pairwise independence.

Let $b = \lceil \log(n+1) \rceil$ and let $X_1, \ldots, X_b$ be independent random bits. As explained above we can obtain $n$ pairwise independent random bits $Y_1, \ldots, Y_n$ from the $X_i$. Now writing $V = \{1, \ldots, n\}$ for the set of vertices of $G$ we take $A = \{i : Y_i = 1\}$ as our random cut. An edge $(i, j)$ crosses the cut iff $Y_i \neq Y_j$, and this happens with probability $1/2$ by pairwise independence. Thus, writing $X$ for the number of edges crossing the cut, we have $\mathbb{E}[X] \geq m/2$. We conclude that there exists a choice of $Y_1, \ldots, Y_n$ yielding a cut of value $\geq m/2$. In turn this implies that there is a choice of $X_1, \ldots, X_b$ yielding a cut of value $\geq m/2$. But now we have a polynomial-time deterministic procedure for finding a cut with value $\geq m/2$: try all $2^b = O(n)$ values of $X_1, \ldots, X_b$ until one generates a cut of value at least $m/2$.

#### Sources

- Mitzenmacher and Upfal, 2nd edition, Sections 15.1.2, 15.1.3

## 2.3 Further applications of linearity of expectations

### 2.3.1 Coupon collector's problem

#### The geometric distribution

Random processes in which we repeat an experiment with probability of success $p$ until we succeed are captured by the geometric distribution.

[3] There are two ways to see this. First, because half of all binary strings of length $k$ have an odd number of 1's. Second by the "principle of deferred decisions" (see the textbook).

**Definition 2.3.1.** A random variable $X$ is geometric with parameter $p$ if it takes values $1, 2, \ldots$ with

$$\mathbb{P}(X = n) = (1-p)^{n-1}p.$$

The following properties follow directly from the definition. The first of these is called the *memoryless* property of the geometric distribution.

**Lemma 2.3.2.** *For a geometric random variable $X$ with parameter $p$ and for $n > 0$,*

$$\mathbb{P}(X = n + k \mid X > k) = \mathbb{P}(X = n)$$
$$\mathbb{E}[X] = \frac{1}{p}$$

### Coupon collector's problem - expectation

In the coupon collector's problem, there are $n$ types of coupons. Each time we select a new coupon, its type is chosen u.a.r. from the set of all $n$ types. Our goal is to collect a coupon of each type. The question is, how many samples do we expect this to take?

Let $X_i$ be the number of samples that it takes to collect the $i$'th type that we get (after we've already collected the $(i-1)$st type). We want to compute $\mathbb{E}[\sum_{i=1}^{n} X_i]$. Note that if we already have $i-1$ types, the probability of obtaining a new type is

$$p_i = \frac{n-i+1}{n}.$$

We repeat the experiment until we succeed, at which point we have $i$ types. Thus, the random variable $X_i$ is geometric with parameter $p_i$. By linearity of expectations, we can now compute the expected number of samples to collect all types:

$$\begin{aligned}
\mathbb{E}[\sum_{i=1}^{n} X_i] &= \sum_{i=1}^{n} \mathbb{E}[X_i] \\
&= \sum_{i=1}^{n} 1/p_i \\
&= \sum_{i=1}^{n} \frac{n}{n-i+1} \\
&= nH_n,
\end{aligned}$$

where $H_n = 1 + \frac{1}{2} + \cdots + \frac{1}{n}$ is the $n$'th *Harmonic number*, approximately equal to $\ln n$.

### Sources

- Mitzenmacher and Upfal, 2nd edition, Section 2.4

### 2.3.2 Randomised quicksort

Quicksort is a simple and practical algorithm for sorting large sequences. The algorithm works as follows: it selects as *pivot* an element $v$ of the sequence and then partitions the sequence into two parts: part $L$ that contains the elements that are smaller than the pivot element and part $R$ that contains the elements that are larger than the pivot element. It then sorts recursively $L$ and $R$ and puts everything together: sorted $L$, $v$, sorted $R$.

Randomised Quicksort selects the pivot elements independently and u.a.r. We want to compute the expected number of comparisons of this algorithm.

**Theorem 2.3.3.** *For every sequence of $n$ elements, the expected number of comparisons made by Randomised Quicksort is $2n \ln n + O(n)$.*

*Proof.* Let $v_1, \ldots, v_n$ be the elements of the sequence *after we sort them*. The important step in the proof is to define indicator random variables $X_{i,j}$, indicating whether $v_i$ and $v_j$ are compared at any point during the execution of the algorithm. We want to estimate $\mathbb{E}[\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} X_{i,j}]$.

The heart of the proof is to compute $\mathbb{E}[X_{i,j}]$ by focusing on the elements $S_{i,j} = \{v_i, \ldots, v_j\}$ and observing that

- the algorithm compares $v_i$ with $v_j$ if and only if the first element from $S_{i,j}$ to be selected as pivot is either $v_i$ or $v_j$.

Given that the pivot elements are selected independently and u.a.r., the probability with which this happens is $2/(j-i+1)$. From this and the linearity of expectations, we get

$$\begin{aligned}
E\left[\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} X_{i,j}\right] &= \sum_{i=1}^{n-1}\sum_{j=i+1}^{n} \frac{2}{j-i+1} \\
&= \sum_{i=1}^{n-1}\sum_{k=2}^{n-i+1} \frac{2}{k} \\
&= \sum_{k=2}^{n}\sum_{i=1}^{n-k+1} \frac{2}{k} \\
&= \sum_{k=2}^{n}(n-k+1)\frac{2}{k} \\
&= (n+1)\sum_{k=2}^{n}\frac{2}{k} - 2(n-1) \\
&= 2(n+1)\sum_{k=1}^{n}\frac{1}{k} - 4n \\
&= 2(n+1)H_n - 4n.
\end{aligned}$$

Since $H_n = \ln n + \Theta(1)$, we get that the expectation is $2n \ln n + \Theta(n)$. □

**Sources**

- Mitzenmacher and Upfal, 2nd edition, Section 2.5

### 2.3.3 Jensen's inequality

Another application of the linearity of expectations is a proof of Jensen's inequality, a very general lemma and an important tool in probability theory.

**Lemma 2.3.4.** *For every convex function $f\colon \mathbb{R} \to \mathbb{R}$ and every random value $X$ with finite expectation:*

$$\mathbb{E}[f(X)] \geq f(\mathbb{E}[X]).$$

The theory of convex functions of one or more variables plays an important part in many fields and in particular in algorithms. A function is called *convex* if for every $x$, $y$, and $\lambda \in [0,1]$: $f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda)f(y)$.

Jensen's inequality holds for every convex function $f$, but for simplicity we will prove it only for differentiable $f$. For differentiable functions, the definition of convexity is equivalent to: for every $x$ and $y$, $f(y) \geq f(x) + f'(x)(y-x)$.

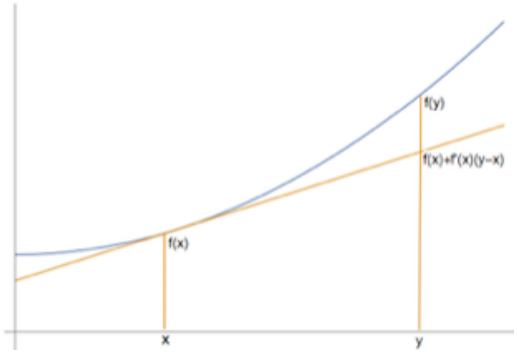



Figure 2.1: A convex function: $f(y) \geq f(x) + f'(x)(y-x)$

With this, Jensen's inequality is a trivial application of the linearity of expectations:

*Proof.* Let $\mu = \mathbb{E}[X]$. We then have $f(X) \geq f(\mu) + f'(\mu)(X-\mu)$. This implies that

$$\begin{aligned}\mathbb{E}[f(X)] &\geq f(\mu) + f'(\mu)(\mathbb{E}[X] - \mu)\\ &= f(\mu) + f'(\mu)(\mu - \mu)\\ &= f(\mu)\\ &= f(\mathbb{E}[X]).\end{aligned}$$

□

If a function is twice-differentiable on an interval and its second derivative is non-negative on the interval then it is convex on the interval. For example, the function $f(x) = x^2$ has $f'(x) = 2x$ and $f''(x) = 2$ so this is convex.

An immediate application of Jensen's inequality is therefore that for every random variable $X$ with finite expectation, $\mathbb{E}[X^2] \geq (\mathbb{E}[X])^2$.

**Sources**

- Mitzenmacher and Upfal, 2nd edition, Section 2.1

PART 3
TAIL BOUNDS

## 3.1 Markov's and Chebyshev's inequalities

We have seen that in many cases we can easily compute exactly or approximately the expectation of random variables. For example, we have computed the expected value of the number of comparisons used by Randomised Quicksort to be $2n \ln n + \Theta(n)$. But this is not very informative about the behaviour of the algorithm. For example, it may be that the running time of the algorithm is either approximately $n$ or approximately $n^2$, but never close to the expected running time. It would be much more informative if we knew the distribution of the number of comparisons used by the algorithm. *Concentration bounds*–also known as *tail bounds*–go a long way towards providing this kind of information: they tell us that a random variable takes values close to its expectation with high probability. We will see a few concentration bounds in this course, such as Markov's and Chebyshev's inequalities and the Chernoff bound.

### 3.1.1 Union bound

Before we turn our attention to concentration bounds, we will consider a simple bound on probabilities, the union bound. In most applications, it is not strong enough to give any useful result, but there are surprisingly many cases where the bound is useful and sometimes it is the only available tool.

**Lemma 3.1.1** (Union bound)**.** *For any countable set of events $E_1, E_2, \ldots$,*

$$\mathbb{P}(\cup_{i \geq 1} E_i) \leq \sum_{i \geq 1} \mathbb{P}(E_i).$$

If the events $E_i$ are mutually exclusive, the union bound holds with equality[1].

#### Example

The Ramsey number $R(k, m)$ is the smallest integer $n$ such that for every graph $G$ of $n$ vertices, either $G$ has a $k$-clique or the complement graph $\overline{G}$ has an $m$-clique. For example $R(3, 3) = 6$. Ramsey's theorem, which can be proved by induction, states that $R(k, m)$ is well defined in the sense that every sufficiently large graph $G$ either has a $k$-clique in $G$ or an $m$-clique in $\overline{G}$. Of particular interest are the numbers $R(k, k)$, and the following theorem, based on the union bound, gets within a constant factor of the best known lower bound on these numbers.

[1] This is one of the probability axioms.

**Theorem 3.1.2.** *If* $\binom{n}{k} < 2^{\binom{k}{2}-1}$ *then* $R(k,k) > n$. *Therefore* $R(k,k) = \Omega(k2^{k/2})$.

*Proof.* Consider the complete graph on $n$ vertices and colour its edges independently and u.a.r. with two colours. If there is a positive probability that there is no monochromatic $k$-clique, then $R(k,k) > n$.

A fixed subset of $k$ vertices is monochromatic with probability $2^{1-\binom{k}{2}}$. There are $\binom{n}{k}$ such subsets, so, by the union bound, the probability that there exists a monochromatic $k$-clique is at most $\binom{n}{k}2^{1-\binom{k}{2}}$. If this is less than 1, there exists a colouring that does not create a monochromatic $k$-clique. Thus, if $\binom{n}{k}2^{1-\binom{k}{2}} < 1$ then $R(k,k) > n$. This proves the first part of the theorem. The second part follows by solving for $n$ and using Sitrling's approximation. □

### 3.1.2 Markov's inequality

Markov's inequality is the weakest concentration bound.

**Theorem 3.1.3** (Markov's inequality)**.** *For every non-negative random variable* $X$ *and every* $a > 0$,

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

*Proof.* The proof explores an idea that we have seen a few times: If $Y$ is a 0-1 indicator variable, then $\mathbb{E}[Y] = \mathbb{P}(Y = 1)$. Now let $Y$ be the indicator variable for the event $X \geq a$. Observe that $Y \leq X/a$ because

- if $Y = 1$ then $X \geq a$
- if $Y = 0$ then $Y \leq X/a$, because $X \geq 0$.

Therefore $\mathbb{P}(X \geq a) = \mathbb{P}(Y = 1) = \mathbb{E}[Y] \leq \mathbb{E}[X/a] = \mathbb{E}[X]/a$, which proves the lemma. □

Although Markov's inequality does not provide much information about the distribution of the random variable $X$, it has some advantages:

- it requires that we only know the expected value $\mathbb{E}[X]$
- it applies to every non-negative random variable $X$
- it is a useful tool to obtain stronger inequalities when we know more about the random variable $X$

**Example:** Markov's inequality is useful for turning bounds on expected running times to probabilistic bounds about running times. For example, the expected number of comparisons of Randomised Quicksort is $2n \ln n + \Theta(n) \leq 3n \ln n$, for large $n$. Therefore, the probability that the number of comparisons is more than $6n \ln n$ is at most $1/2$; more generally, the probability that the number of comparisons is more than $3kn \ln n$ is at most $1/k$, for every $k \geq 1$.

**Example:** Let's try to use Markov's inequality to bound the probability of success of the randomised Max-Cut algorithm. We have seen that the value $X$ of the cut returned by the algorithm has expectation $m/2$. What is the probability that the returned cut has value

at least $m/4$? By applying Markov's inequality, we get that the probability is at most 2, a useless bound!

But here is a trick: let's consider the number of edges that *do not* cross the cut. This is a random variable $Y = m - X$ whose expectation is also $m/2$. We want to bound the probability $\mathbb{P}(X \geq m/4) = \mathbb{P}(Y \leq 3m/4) = 1 - \mathbb{P}(Y > 3m/4)$. By Markov's inequality $\mathbb{P}(Y > 3m/4) \leq 2/3$, from which we get that the probability that the randomised Max-Cut algorithm returns a cut of value $X \geq m/4$ is at least $1 - 2/3 = 1/3$. By a straightforward generalisation, we get that the probability that the randomised Max-Cut algorithm returns a cut of value greater than $(1-\varepsilon)m/2$ is at least $\varepsilon/(1+\varepsilon)$. In particular, by taking $\varepsilon = 1/m$ and using the fact that $m$ is an integer, we get that the probability that the algorithm returns a cut of value at least $m/2$ (i.e., greater than $(m-1)/2$) is at least $1/(m+1)$.

The technique of applying Markov's inequality to a function of the random variable that we are interested in, instead of to the random variable itself, is very powerful and we will use it repeatedly.

### 3.1.3 Variance and moments

**Definition 3.1.4.** The $k$'th moment of a random variable $X$ is $\mathbb{E}[X^k]$.

**Definition 3.1.5.** The variance of a random variable $X$ is

$$\mathbb{V}\mathrm{ar}[X] = \mathbb{E}[(X - \mathbb{E}[X])^2] = \mathbb{E}[X^2] - \mathbb{E}[X]^2.$$

Its square root is called standard deviation: $\sigma[X] = \sqrt{\mathbb{V}\mathrm{ar}[X]}$.

The second equality holds because

$$\mathbb{E}[(X - \mathbb{E}[X])^2] = \mathbb{E}[X^2 - 2X\,\mathbb{E}[X] + \mathbb{E}[X]^2] = \mathbb{E}[X^2] - 2\,\mathbb{E}[X]\,\mathbb{E}[X] + \mathbb{E}[X]^2 = \mathbb{E}[X^2] - \mathbb{E}[X]^2.$$

**Definition 3.1.6.** The covariance of two random variables $X$ and $Y$ is

$$\mathbb{C}\mathrm{ov}[X, Y] = \mathbb{E}[(X - \mathbb{E}[X])(Y - E[Y])].$$

We say that $X$ and $Y$ are positively (negatively) correlated when their covariance is positive (negative).

**Lemma 3.1.7.** *For any random variables $X$ and $Y$:*

$$\mathbb{V}\mathrm{ar}[X + Y] = \mathbb{V}\mathrm{ar}[X] + \mathbb{V}\mathrm{ar}[Y] + 2\,\mathbb{C}\mathrm{ov}[X, Y].$$

*If $X$ and $Y$ are independent then $\mathbb{C}\mathrm{ov}[X, Y] = 0$ and $\mathbb{V}\mathrm{ar}[X + Y] = \mathbb{V}\mathrm{ar}[X] + \mathbb{V}\mathrm{ar}[Y]$.*

*More generally, for pairwise independent random variables $X_1 \ldots, X_n$:*

$$\mathbb{V}\mathrm{ar}\left[\sum_{i=1}^{n} X_i\right] = \sum_{i=1}^{n} \mathbb{V}\mathrm{ar}[X_i].$$

### 3.1.4 Chebyshev's inequality

Chebyshev's inequality is a tail bound that involves the first two moments.

**Theorem 3.1.8** (Chebyshev's inequality)**.** *For every random variable $X$ and every $a > 0$,*

$$\mathbb{P}(|X - \mathbb{E}[X]| \geq a) \leq \frac{\mathbb{V}\mathrm{ar}[X]}{a^2}.$$

*Proof.* The proof is an application of Markov's inequality to the (non-negative) random variable $Y = (X - \mathbb{E}[X])^2$. Indeed, from the definition of $\mathbb{V}\mathrm{ar}[X]$ we have that $\mathbb{E}[Y] = \mathbb{V}\mathrm{ar}[X]$, and by applying Markov's inequality we get

$$\mathbb{P}((X - \mathbb{E}[X])^2 \geq a^2) \leq \frac{\mathbb{E}[Y]}{a^2} = \frac{\mathbb{V}\mathrm{ar}[X]}{a^2},$$

which proves the theorem. □

#### Example

We have seen that indicator random variables are very useful. An indicator random variable represents the outcome of a *Bernoulli trial*, which is a random experiment with two outcomes, success or failure.

A *binomial random variable* $X$ with parameters $n$ and $p$ is the sum of $n$ independent Bernoulli (indicator) random variables, each with probability $p$; we write $X \sim B(n, p)$. It follows immediately that for $X \sim B(n, p)$,

$$\mathbb{P}(X = i) = \binom{n}{i} p^i (1-p)^{n-i}.$$

The expectation and variance of a binomial random variable $X$ with parameters $n$ and $p$ is $np$ and $np(1-p)$, respectively.

We often need to estimate the probability $\mathbb{P}(X \geq k) = \sum_{i=k}^{n} \mathbb{P}(X = i) = \sum_{i=k}^{n} \binom{n}{i} p^i (1-p)^{n-i}$. Instead of this complicated sum, we want a simple expression that approximates it. Let's see what we get when we apply Markov's and Chebyshev's inequality to bound $\mathbb{P}(X \geq 3n/4)$ when $p = 1/2$ (equivalent to the probability of getting more than $3n/4$ heads when we throw a fair coin $n$ times).

#### Markov's inequality

$$\mathbb{P}(X \geq 3n/4) \leq (n/2)/(3n/4) = 2/3$$

#### Chebyshev's inequality

$$\mathbb{P}(X \geq 3n/4) \leq \mathbb{P}(|X - n/2| \geq n/4) \leq (n/4)/(n/4)^2 = 4/n.$$

Markov's and Chebyshev's inequalities provide some bounds on this probability, but they are not very accurate. We will see later that a Chernoff bound gives a much better estimate.

### 3.1.5 Coupon collector's problem - probability

In many applications, Chebyshev's inequality is sufficient to get decent results.

Consider the coupon collector's problem. We have seen that the time for collecting all coupons is the sum of $n$ geometric variables $X_i$, $i = 1, \dots, n$, with parameters $(n - i + 1)/n$ and $\mathbb{E}[\sum_{i=1}^{n} X_i] = nH_n$.

The variance of a geometric variable with parameter $p$ is $(1 - p)/p^2$. Given that the $X_i$'s are independent, we have

$$\begin{aligned}
\mathbb{V}\mathrm{ar}\left[\sum_{i=1}^{n} X_i\right] &= \sum_{i=1}^{n} \frac{n(i-1)}{(n-i+1)^2} \\
&\leq \sum_{i=1}^{n} \left(\frac{n}{n-i+1}\right)^2 \\
&= n^2 \sum_{i=1}^{n} \frac{1}{i^2} \\
&\leq \frac{\pi^2 n^2}{6}.
\end{aligned}$$

With this, we can apply Chebyshev's inequality to get

$$\mathbb{P}(|X - nH_n| \geq nH_n) \leq \frac{\pi^2 n^2/6}{(nH_n)^2} = O\left(\frac{1}{(\log n)}^2\right),$$

which says that the probability to exceed the expectation by a factor of 2 is small. Although Chebyshev's inequality provide a decent estimate in this case, we know much stronger concentration bounds for the coupon collector's problem.

**Sources**

- Mitzenmacher and Upfal, 2nd edition, Sections 3.1–3.3
- Nick Harvey, Lecture2Notes.pdf (page 2)
- Lap Chi Lau, L02.pdf (page 4)

## 3.2 Chernoff bounds

### 3.2.1 Chernoff bounds

One of most useful concentration bounds is Chernoff's bound for binomial random variables and more generally for sums of $0 - 1$ *independent* random variables.

**Theorem 3.2.1** (Chernoff bounds)**.** *Let $X_1, \dots, X_n$ be 0-1 independent random variables such that $\mathbb{P}(X_i = 1) = p_i$. Let $X = \sum_{i=1}^{n} X_i$ be their sum and $\mu = E[X]$ its expected value. Then for $\delta > 0$,*

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

*and for* $0 < \delta < 1$,

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

*Proof.* We will only prove the first inequality, as the proof of the second inequality is very similar. The proof is based on applying Markov's inequality to the random variable $e^{tX}$, for some appropriate parameter $t > 0$.

$$\begin{aligned}
\mathbb{P}(X \geq (1+\delta)\mu) &= \mathbb{P}(e^{tX} \geq e^{t(1+\delta)\mu}) \\
&\leq \frac{\mathbb{E}[e^{tX}]}{e^{t(1+\delta)\mu}} \\
&= \frac{\mathbb{E}[e^{t\sum_{i=1}^{n} X_i}]}{e^{t(1+\delta)\mu}} \\
&= \frac{\prod_{i=1}^{n} \mathbb{E}[e^{tX_i}]}{e^{t(1+\delta)\mu}} \\
&= \frac{\prod_{i=1}^{n} (p_i e^t + (1-p_i))}{e^{t(1+\delta)\mu}}.
\end{aligned}$$

The above inequality comes from Markov's inequality, and the second-to-last equality follows from the fact that the $X_i$'s are independent. These are the two main ingredients of the proof; everything else in the proof consists of standard algebraic manipulations and elementary calculus.

Using the fact that for every $x$, $e^x \geq 1 + x$, we compute

$$\begin{aligned}
p_i e^t + (1-p_i) &= 1 + p_i(e^t - 1) \leq e^{p_i(e^t-1)}, \\
\prod_{i=1}^{n} (p_i e^t + (1-p_i)) &\leq \prod_{i=1}^{n} e^{p_i(e^t-1)} = e^{\sum_{i=1}^{n} p_i(e^t-1)} = e^{\mu(e^t-1)}
\end{aligned}$$

Therefore, for every $t > 0$,

$$\mathbb{P}(X \geq (1+\delta)\mu) \leq \frac{e^{\mu(e^t-1)}}{e^{t(1+\delta)\mu}} = \left(\frac{e^{e^t-1}}{e^{t(1+\delta)}}\right)^{\mu}.$$

We get the first inequality by choosing $t = \ln(1+\delta) > 0$. □

The next corollary gives slightly weaker bounds that are simpler to apply. They can be derived directly from the theorem using the following inequalities for $0 < \delta < 1$, which can be established by elementary calculus techniques:

$$\begin{aligned}
\delta - (1+\delta)\ln(1+\delta) &\leq -\delta^2/3, \\
-\delta - (1-\delta)\ln(1-\delta) &\leq -\delta^2/2.
\end{aligned}$$

**Corollary 3.2.2.** *Let $X_1, \ldots, X_n$ be 0-1 independent random variables such that $\mathbb{P}(X_i = 1) = p_i$. Let $X = \sum_{i=1}^{n} X_i$ and $\mu = E[X]$. Then for $0 < \delta < 1$,*

$$\begin{aligned}
\mathbb{P}(X \geq (1+\delta)\mu) &\leq e^{-\mu\delta^2/3}, \\
\mathbb{P}(X \leq (1-\delta)\mu) &\leq e^{-\mu\delta^2/2}.
\end{aligned}$$

### 3.2.2 Examples

#### Coin flips

Let's analyse the number of heads that come up in $n$ independent fair coin flips. Let $X_i$ be the indicator random variable for the $i$'th coin flip and let $X = \sum_{i=1}^{n} X_i$ be the total number of heads. Applying the Chernoff bound we get

$$\mathbb{P}\left(X - \frac{n}{2} \geq \frac{1}{2}\sqrt{6n \ln n}\right) \leq \exp\left(-\frac{1}{3}\frac{n}{2}\frac{6 \ln n}{n}\right) = \frac{1}{n}.$$

There is a similar bound for $\mathbb{P}(\frac{n}{2} - X \geq \frac{1}{2}\sqrt{6n \ln n})$. This shows that the number of heads is tightly concentrated around the mean. The probability of being away from the mean drops exponentially. For example,

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

Compare this with the bounds that we got using the Markov's and Chebyshev's inequalities in Section 3.1:

#### Markov's inequality

$$\mathbb{P}(X \geq 3n/4) \leq (n/2)/(3n/4) = 2/3$$

#### Chebyshev's inequality

$$\mathbb{P}(X \geq 3n/4) \leq \mathbb{P}(|X - n/2| \geq n/4) \leq (n/4)/(n/4)^2 = 4/n.$$

#### Random walk on a line

In many applications, we consider sums of random variables that take values $\{-1, 1\}$ instead of $\{0, 1\}$. We can translate the above Chernoff bounds to this case, but we can also just use the same technique to get similar bounds:

**Theorem 3.2.3.** *Let $X_1, \ldots, X_n$ be independent random variables with $\mathbb{P}(X_i = 1) = \mathbb{P}(X_i = -1) = 1/2$, and let $X = \sum_{i=1}^{n} X_i$. Then for $a > 0$,*

$$\mathbb{P}(X \geq a) \leq e^{-a^2/2n}.$$

Consider a particle that takes an unbiased random walk on a line: it starts at $Z_0 = 0$ and at each time step $t$ it moves from its current position $Z_{t-1}$ either left $Z_t = Z_{t-1} - 1$ or right $Z_t = Z_{t-1} + 1$ with equal probability.

We will use Chernoff bounds to show that after $n$ steps its position is with high probability in distance $O(\sqrt{n \ln n})$ from the origin. Indeed $X_i = Z_i - Z_{i-1}$ are independent random variables with $\mathbb{P}(X_i = 1) = \mathbb{P}(X_i = -1) = 1/2$, and $Z_i = \sum_{i=1}^{n} X_i$. Therefore

$$\mathbb{P}(|Z_n| \geq \sqrt{cn \ln n}) \leq 2e^{-cn \ln n/2n} = 2n^{-c/2}.$$

For $c = 2$, the probability of being more than $\sqrt{2n \ln n}$ steps away from the origin is at most $2/n$.

*Some useful bounds that we often use:*

- $1 + x \leq e^x$, *for every* $x$.
  *Proof (we did this in a footnote in Section 1.2): Let* $f(x) = e^x - 1 - x$. *Then* $f'(x) = e^x - 1$, *which is* $0$ *at* $x = 0$. $f''(x) = e^x > 0$ *so* $f(x)$ *is minimised at* $x = 0$. *Thus,* $f(x) \geq f(0) = 0$.
- $\frac{e^x + e^{-x}}{2} \leq e^{x^2/2}$
  *Proof: Consider the Taylor expansions of* $e^x$ *and* $e^{-x}$:

$$\begin{aligned}(e^x + e^{-x})/2 &= \left(\sum_{k=0}^{\infty} x^k/k! + \sum_{k=0}^{\infty} (-x)^k/k!\right)/2 \\ &= \sum_{k=0}^{\infty} x^{2k}/(2k)! \\ &\leq \sum_{k=0}^{\infty} (x^2/2)^k/k! \\ &= e^{x^2/2}.\end{aligned}$$

  *The inequality follows from term-wise comparison:* $(2k)! \geq 2^k k!$.
- $\left(\frac{n}{k}\right)^k \leq \binom{n}{k} \leq \left(\frac{en}{k}\right)^k$
  *Proof: For the lower bound,*

$$\binom{n}{k} = \frac{n(n-1)\cdots(n-k+1)}{k(k-1)\cdots 1} = \frac{n}{k} \cdots \frac{n-k+1}{1} \geq \frac{n}{k} \cdots \frac{n}{k} = \left(\frac{n}{k}\right)^k$$

  *For the upper bound,*

$$\binom{n}{k} = \frac{n(n-1)\cdots(n-k+1)}{k(k-1)\cdots 1} \leq \frac{n^k}{k!} \leq \frac{n^k}{k^k/e^k}.$$

  *To see the last inequality, take the Taylor expansion of* $e^k$ *and observe that its* $k$*'th term is* $k^k/k!$, *which shows* $e^k \geq k^k/k!$.

**Sources**

- Mitzenmacher and Upfal, 2nd edition, Sections 4.1-4.3
- Lap Chi Lau, L03.pdf (page 1)

## 3.3 Applications of Chernoff bounds

### 3.3.1 Set Balancing

Let $A$ be an $n \times m$ matrix with entries in $\{0, 1\}$. Columns represent *individuals* and rows represent *features*, and $A_{i,j}$ indicates whether individual $j$ has feature $i$. The *set balancing problem* asks to partition the set of individuals into two classes such that each feature is as balanced as possible between the classes. For example, one might want to test a new drug and a placebo respectively on two groups that are as similar as possible.

We represent the partition by an $m$-dimensional vector $b$ with entries in $\{-1, 1\}$, where the value of $b_i$ indicates the class to which individual $i$ belongs. Our goal is to minimise the *discrepancy* which is the maximum entry (in absolute value) in the vector $A \cdot b$, viewing $b$ as a column vector. If $c$ is the vector such that $A \cdot b = c$ then the discrepancy is written as follows: $||A \cdot b||_\infty = \max_{i \in \{1,\ldots,n\}} |c_i|$. It is the maximum, over all features, of the difference between the respective numbers of people possessing the feature in the two groups.

Here is a simple randomised algorithm for selecting the vector $b$:

*Independently choose each entry $b_j$ to be 1 or $-1$, each with probability $1/2$.*

Note that this procedure ignores the entries of the feature matrix $A$.

**Theorem 3.3.1.** *For a random vector $b$ with entries chosen independently and equiprobably from $\{-1, 1\}$, we have $\mathbb{P}(||A \cdot b||_\infty \geq \sqrt{4m \ln n}) \leq 2/n$.*

*Proof.* We first focus on a specific row $a_i$ of the matrix $A$. We have two cases depending on the number of non-zero entries $k$ of $a_i$:

- If $k = 0$ (nobody has feature $i$) then $|a_i \cdot b| = 0 \leq \sqrt{4m \ln n}$.
- Otherwise, we apply a Chernoff bound (Theorem 3.2.3)

$$\mathbb{P}(|a_i \cdot b| \geq \sqrt{4m \ln n}) \leq 2e^{-4m \ln n/(2k)} \leq 2e^{-4m \ln n/(2m)} = 2n^{-2}.$$

Since there are $n$ rows, the theorem follows from the union bound.

□

### 3.3.2 Balls and bins - maximum load

We throw $n$ balls into $n$ bins and we are interested in the maximum load.

First let's consider a fixed bin, say bin 1. Its expected load is 1. Since the balls are thrown independently, we can use the Chernoff bounds. Here we have the sum of $n$ 0-1 independent random variables, each with probability $1/n$. In other words, the number of balls into bin 1 is $Y_1 \sim B(n, 1/n)$. A Chernoff bound (Theorem 3.2.1) gives (with $\mu = \mathbb{E}[Y_1] = 1$)

$$\mathbb{P}(Y_1 \geq 1 + \delta) \leq \frac{e^\delta}{(1+\delta)^{1+\delta}}.$$

If we select $1 + \delta = 3 \ln n / \ln \ln n$, the right-hand side is at most $1/n^2$.

By the union bound, the probability that there is a bin with load at least $1 + \delta$ is at most $n(1/n^2) = 1/n$. So we have proved the following theorem.

**Theorem 3.3.2.** *When we throw $n$ balls into $n$ bins uniformly and independently, the probability that the maximum load is more than $3 \ln n / \ln \ln n$ is at most $1/n$, for $n$ sufficiently large.*

To bound the expected maximum load (for sufficiently large $n$), we note the following:

- with probability at least $1 - 1/n$, the maximum load is at most $3 \ln n / \ln \ln n$, and
- the maximum load is at most $n$.

Therefore the expected maximum load is at most

$$\mathbb{P}(\max \leq 3\log n/\log\log n)3\ln n/\ln\ln n + \mathbb{P}(\max > 3\log n/\log\log n)n$$

which is at most

$$(1-1/n)\cdot 3\ln n/\ln\ln n + (1/n)\cdot n \leq 1 + 3\ln n/\ln\ln n.$$

This is an upper bound on the expected maximum load and it is tight. One can derive the opposite direction using the *second moment method*, but this is beyond the scope of this course. The following theorem is offered without proof.

**Theorem 3.3.3.** *When we throw $n$ balls into $n$ bins uniformly and independently, the expected maximum load is $\Theta(\log n/\log\log n)$.*

In fact, we know of very precise bounds, that we will also not prove here. The expected maximum load is $(1+o(1))\ln n/\ln\ln n$.

**Sources**

- Mitzenmacher and Upfal, 2nd edition, sections 4.4, 5.2.1

### 3.3.3 Randomness and non-uniformity

We consider here an application of Chernoff bounds to computational complexity. Recall the definition of the complexity class BPP.

**Definition 3.3.4.** A language $L$ is in Bounded-error Probabilistic-Polynomial-time (BPP) if there is a polynomial-time deterministic algorithm $A(x,r)$ whose bit sequence $r$ has length $p(|x|)$, for some polynomial $p(\cdot)$, and for every $x$:

- if $x \in L$, $A(x,r)$ accepts for at least $3/4$ of the $r$'s
- if $x \notin L$, $A(x,r)$ rejects for at least $3/4$ of the $r$'s.

We can think of $A(x,r)$ as having value $1$ for accept and $0$ for reject. We noted in Section 1.2 that we can bring the probability of getting the correct answer very close to $1$ by repeating the algorithm many times and taking the majority. Here is a detailed "powered" algorithm for BPP, which runs algorithm $A$ $k$ times and takes the majority.

- *Input $x$*
- *Choose random bit sequences $r[1],\ldots,r[k]$*
- *If $\sum_1^k A(x,r[i]) \geq k/2$ Return 1 Else Return 0*

We can use a Chernoff bound to prove the following theorem.

**Theorem 3.3.5.** *For any $x$, the probability that the powered algorithm is incorrect is at most $e^{-k/24}$.*

*Proof.* Suppose $x \in L$. $A(x, r[i])$ is a Bernoulli trial with success probability $p \geq 3/4$, and $X = \sum_{i=1}^{k} A(x, r[i])$ is a Binomial random variable with distribution $B(k, p)$. Let $\mu = \mathbb{E}[X] = pk$. Let $\delta = 1 - 1/(2p)$ so that $k/2 = (1-\delta)\mu$. Since $p \geq 3/4$, we have $\delta \geq 1/3$. By a Chernoff bound (Corollary 3.2.2),

$$\mathbb{P}(X \leq k/2) = \mathbb{P}(X \leq (1-\delta)\mu) \leq e^{-\mu\delta^2/2} \leq e^{-\frac{3}{4}k\left(\frac{1}{3}\right)^2\frac{1}{2}} = e^{-k/24}.$$

The argument is similar when $x \notin L$. □

So we could have defined BPP as follows.

**Definition 3.3.6.** A language $L$ is in Bounded-error Probabilistic-Polynomial-time (BPP) if there is a polynomial-time deterministic algorithm $\widehat{A}(x, \hat{r})$ whose bit sequence $\hat{r}$ has length $\hat{p}(|x|)$, for some polynomial $\hat{p}(\cdot)$, and for every $x$:

- if $x \in L$, $\widehat{A}(x, \hat{r})$ accepts for at least $1 - 2^{-(|x|+1)}$ of the $\hat{r}$'s
- if $x \notin L$, $\widehat{A}(x, \hat{r})$ rejects for at least $1 - 2^{-(|x|+1)}$ of the $\hat{r}$'s.

How do we get the algorithm $\widehat{A}$ in Definition 3.3.6? We just take the algorithm $A$ from Definition 3.3.4, let $k = 24(|x| + 1)$ and use the powered algorithm. The random string $\hat{r}$ is the concatenation of $r[1], \ldots, r[k]$. Thus, $\hat{p}(|x|) = 24(|x| + 1)p(|x|)$. By Theorem 3.3.5, the probability that $\widehat{A}(x, \hat{r})$ is incorrect is at most $e^{-k/24} \leq 2^{-(|x|+1)}$.

This section is about "non-uniformity", and what "non-uniformity" means is that we are willing to build a special algorithm just for inputs of size-$n$ (you can think about implementing it as a circuit with $n$ bits). The following theorem shows that the circuit will not need randomness.

**Lemma 3.3.7.** *If $L$ is in BPP then for every $n$ there is a bit sequence $\hat{r}$ of length $\hat{p}(n)$ such that for every length-$n$ input $x$, $\widehat{A}(x, \hat{r})$ is correct.*

*Proof.* Let's choose $\hat{r}$ uniformly from bit sequences of length $\hat{p}(n)$. Then the probability (over the choice of $\hat{r}$) that there exists a length-$n$ $x$ such that $\widehat{A}(x, \hat{r})$ is incorrect is at most $\sum_x \mathbb{P}_{\hat{r}}(\widehat{A}(x, \hat{r}) \text{ is incorrect})$. But there are at most $2^n$ length-$n$ inputs $n$, and each probability is at most $2^{-(n+1)}$, so the total is at most $1/2$.

We conclude that the probability (over the choice of $\hat{r}$) that $\widehat{A}(x, \hat{r})$ gets the right answer for all length-$n$ inputs $x$ is at least $1/2$.

This means that there exists some $\hat{r}$ such that $\widehat{A}(x, \hat{r})$ is correct on every length-$n$ input $x$. □

Thus, we have the following theorem.

**Theorem 3.3.8.** *If $L$ is in BPP, then there is a polynomial $q$ so that, for every input length $n$, there is a circuit of size at most $q(n)$ that decides $L$ for all inputs of size $n$.*

**Sources**

- Luca Trevisan's notes lecture08.pdf

PART 4
MARKOV CHAINS

## 4.1 Markov chains: 2SAT and 3SAT

### 4.1.1 Markov chains

A finite discrete-time *Markov chain* is described by a finite set $\Omega$ of states together with a transition matrix $P$ whose rows and columns are indexed by $\Omega$. The transition matrix is *row-stochastic* which means that the entries are non-negative real numbers, and the entries in each row sum to 1.

A *Markov chain* described by $\Omega$ and $P$ is a sequence $X = X_0, X_1, X_2, \ldots$ of random variables such that, for all positive integers $t$ and all sequences $a_0, \ldots, a_t$ of states in $\Omega$,

$$\mathbb{P}(X_t = a_t \mid X_{t-1} = a_{t-1}, \ldots, X_0 = a_0) = \mathbb{P}(X_t = a_t \mid X_{t-1} = a_{t-1}) = P_{a_{t-1},a_t}.$$

The variable $X_t$ represents the state of the Markov chain at time $t$, and the matrix $P$ encodes the state-to-state transition probabilities. These can also be represented by a directed edge-weighted graph whose vertices are the states in $\Omega$ and whose edges give transition probabilities (omitting transitions that happen with probability 0). We refer to this as the "underlying graph" of the Markov chain.

The characteristic property of a Markov chain (called the Markov property) is that the distribution of $X_t$ depends only on $X_{t-1}$. The value of $X_t$ may depend on any or all of the random variables $X_1, \ldots, X_{t-1}$, but this dependence is completely captured by $X_{t-1}$.

In addition to considering finite Markov chains we could consider Markov chains where the state space is countable (so the states are in bijection with the natural numbers). There are even Markov chains where $\Omega$ is uncountable, but we won't consider these.

The Markov chains that we study are time-homogeneous, which means that the probability $\mathbb{P}(X_t = a_t \mid X_{t-1} = a_{t-1})$ is independent of $t$.

We can use the transition matrix to compute the probability that the chain is in a given state at a given time. Suppose $\Omega = \{0, \ldots, n\}$. Let $p_i(t) = \mathbb{P}(X_t = i)$ and $\overline{p}(t) = (p_0(t), \ldots, p_n(t))$. Then $p_i(t) = \sum_{j=0}^{n} p_j(t-1)P_{j,i}$ which is equivalent to $\overline{p}(t) = \overline{p}(t-1)P$.

For any $m \geq 0$ and $t$, consider $P^m_{i,j} = \mathbb{P}(X_{t+m} = j \mid X_t = i)$ — the probability of moving from state $i$ to state $j$ in exactly $m$ steps. For $m = 0$, this is 1 if $i = j$ and 0 otherwise. For $m > 0$, this is equal to $P^m_{i,j} = \sum_{k=0}^{n} P_{i,k}P^{m-1}_{k,j}$. Note that $P^m_{i,j}$ is exactly the $(i,j)$'th entry of the $m$'th power of $P$ (where the 0'th power of a matrix is the identity matrix). so the $m$'th power of $P$ captures the $m$-step transition probabilities.

Thus, $\overline{p}(t+m) = \overline{p}(t)P^m$. Also,

$$\mathbb{P}(X_t = j) = \sum_{i=1}^{n} P^t_{i,j} \cdot \mathbb{P}(X_0 = i).$$

In particular, a Markov chain is completely determined by the distribution of $X_0$ and the transition matrix $P$.

### 4.1.2 Randomised 2-SAT

The 2-SAT problem is the special case of the satisfiability problem: given a 2-CNF formula, either find a satisfying assignment or determine that no such assignment exists. It is known that 2-SAT is solvable in polynomial time, in fact linear time. Here we give a very simple polynomial-time randomised algorithm for 2-CNF formulas.

*Randomised 2-SAT ($\varphi$)*

- *Pick a random assignment*
- *Repeat $2n^2$ times or until a satisfying assignment is found*
  - *Pick an unsatisfied clause $C$*
  - *Pick a literal in $C$ u.a.r., and flip its value*
- *Return current assignment if satisfying, else return "unsatisfiable"*

The 2-SAT algorithm has one-sided errors—if the input $\varphi$ is not satisfiable then the algorithm certainly returns "unsatisfiable", however if $\varphi$ is satisfiable then the algorithm might not find a satisfying assignment. Below we analyse the probability that the algorithm gives the correct answer.

Consider a run of the 2-SAT algorithm in the case that the input $\varphi$ is satisfiable. Let $S$ be a particular assignment that satisfies $\varphi$. We will bound the probability that the algorithm finds $S$. If $\varphi$ is not satisfied by an assignment $A$ there must be an unsatisfied clause $C$. The crucial observation is that one or both literals of $C$ must then have different truth values under $A$ and $S$. Thus if we flip a random literal in $C$ then the probability that we increase the agreement between $A$ and $S$ is either $1/2$ or $1$. Writing $A_t$ for the assignment after $t$ iterations of the main loop and $X_t$ for the number of variables in which $A_t$ agrees with $S$, we have

$$\begin{aligned}
\mathbb{P}(X_{t+1} = 1 \mid X_t = 0) &= 1 \\
\mathbb{P}(X_{t+1} = i+1 \mid X_t = i) &\geq 1/2, && i \in \{1, \ldots, n-1\} \\
\mathbb{P}(X_{t+1} = i-1 \mid X_t = i) &\leq 1/2, && i \in \{1, \ldots, n-1\} \\
\mathbb{P}(X_{t+1} = n \mid X_t = n) &= 1
\end{aligned}$$

If we replace the inequalities with equalities then $X = X_0, X_1, \ldots$ becomes a Markov chain on the state space $\{0, 1, \ldots, n\}$. The expected time that this Markov chain takes to reach state $n$ is greater than or equal to the expected time that the original process takes to reach state $n$, so we will study the Markov chain.

This Markov chain is a "random walk" on the path with vertices $\{0, \ldots, n\}$. The chain always moves to state $1$ from state $0$ — this is called a "reflecting barrier". Once it gets to state $n$, it stays there — this is called an "absorbing barrier". From any other vertex, it picks

between the neighbours with equal probability and goes there.[1]

The expected time until the Markov chain reaches $X_t = n$ is an upper bound on the expected number of steps for the 2-SAT algorithm to find $S$. In turn this is an upper bound for the algorithm to find *some* satisfying assignment.

Let $h_j$ be the expected time for $X$ to first reach $n$ starting at $j$. Then we have

$$\begin{aligned} h_n &= 0 \\ h_0 &= 1 + h_1 \\ h_j &= 1 + \frac{h_{j-1}}{2} + \frac{h_{j+1}}{2}, \qquad j \in \{1, \ldots, n-1\}. \end{aligned}$$

This is a linear system of equations in $n+1$ unknowns. One way to solve the equations is to first find a relationship between $h_j$ and $h_{j+1}$ for $0 \leq j < n$. Here it is not difficult to deduce that $h_j = h_{j+1} + 2j + 1$. Combining this with the boundary condition $h_n = 0$ we get that $h_j = n^2 - j^2$. We conclude that $h_j \leq n^2$ for all $j$.

Let $T$ denote the number of steps for the 2-SAT algorithm to find a satisfying assignment starting from some arbitrary assignment. By Markov's inequality we have

$$\mathbb{P}(T \geq 2n^2) \leq \frac{\mathbb{E}[T]}{2n^2} \leq \frac{n^2}{2n^2} = \frac{1}{2}.$$

**Theorem 4.1.1.** *If $\varphi$ is satisfiable then the 2-SAT algorithm returns a satisfying assignment with probability at least $1/2$. If $\varphi$ is unsatisfiable then the 2-SAT algorithm is always correct.*

### 4.1.3 Randomised 3-SAT

Here we extend the randomised algorithm for 2-SAT to handle 3-SAT—the satisfiability problem for 3-CNF formulas. We find that the expected running time is no longer polynomial in the number of variables $n$. This is not surprising since 3-SAT is NP-complete. The algorithm we describe has expected running time that is exponential in $n$ but is nevertheless significantly better than the running time $O(2^n)$ arising from exhaustive search.

Let us first consider what happens if we try to directly apply the 2-SAT algorithm to the 3-SAT problem. Suppose that a given 3-CNF formula $\varphi$ that has a satisfying assignment $S$. Let $A_t$ be the assignment after $t$ steps and let $X_t$ denote the number of variables in which $A_t$ agrees with $S$. Following the same reasoning as in the 2-SAT case we have that

$$\begin{aligned} \mathbb{P}(X_{t+1} = j+1 \mid X_t = j) &\geq 1/3 \\ \mathbb{P}(X_{t+1} = j-1 \mid X_t = j) &\leq 2/3. \end{aligned}$$

As with the 2-SAT algorithm, we can turn the process $X$ into a Markov chain by changing the above inequalities into equalities, and the expected time for the resulting chain to reach state $n$ is an upper bound for the expected time of the algorithm to find a satisfying assignment.

Again we have a one-dimensional random walk. But notice that in this case the particle is twice as likely to move away from state $n$ as it is to move toward state $n$. This greatly

[1] The term *random walk* usually refers to the random process in which a particle moves on the vertices of a graph, selecting the next vertex u.a.r. from the adjacent vertices. A *lazy random walk* is similar but with $P_{i,i} = 1/2$.

affects the expected time to reach the state $n$. Let us write $h_j$ for the expected number of steps to reach $n$ when starting from state $j$. Then we have the following system of equations:

$$\begin{aligned} h_n &= 0 \\ h_0 &= 1 + h_1 \\ h_j &= 1 + \frac{2h_{j-1}}{3} + \frac{h_{j+1}}{3}\,, \qquad 1 \le j \le n-1 \end{aligned}$$

Again this is a linear system in $n+1$ unknowns, and we solve by first seeking a relationship between $h_j$ and $h_{j+1}$. Here we get that

$$h_j = h_{j+1} + 2^{j+2} - 3$$

for all $j$. With the boundary condition $h_n = 0$ this leads to the solution

$$h_j = 2^{n+2} - 2^{j+2} - 3(n-j)\,.$$

The above bound shows that the expected number of flips to find $S$ is at most $O(2^n)$. This bound is useless since we can find $S$ in $2^n$ steps by exhaustive search. However we can significantly improve the performance of this procedure by incorporating *random restarts* into our search. A random truth assignment leads to a binomial distribution over the states of the Markov chain $X$ centred around the state $n/2$. Since the Markov chain has a tendency to move towards state 0, after running it for a little while we are better off restarting (with a random assignment) than continuing to flip literals. After a random restart, by Chernoff bounds, the current assignment differs from $S$ in approximately $n/2$ positions with high probability.

The algorithm below tries to find a satisfying assignment in $N = O((4/3)^n\sqrt{n})$ phases, where each phase starts with a random restart and consists of $3n$ literal flips. The exact constant $N$ will be determined by the analysis below.

*Randomised 3-SAT ($\varphi$)*

- *Repeat $N$ times or until a satisfying assignment is found*
  - *Pick a random assignment*
  - *Repeat $3n$ times or until a satisfying assignment is found*
    - *Pick an unsatisfied clause $C$*
    - *Pick a literal in $C$ u.a.r., and flip its value*
- *Return current assignment if satisfying, else return "unsatisfiable"*

**Theorem 4.1.2.** *If $\varphi$ is satisfiable then the 3-SAT algorithm returns a satisfying assignment with probability at least $1/2$. If $\varphi$ is unsatisfiable then the algorithm is always correct.*

*Proof.* We first analyse a single phase of the algorithm. Recall that this can be modelled as a one-dimension random walk on the interval $\{0, 1, \ldots, n\}$ in which the probability to move left is $2/3$ and the probability to move right is $1/3$. Let $q_j$ be the probability to reach position $n$ within $3n$ steps starting from position $n - j$. Then we have

$$q_j \ge \binom{3j}{j} \left(\frac{2}{3}\right)^j \left(\frac{1}{3}\right)^{2j}$$

since this is the probability to make exactly $2j$ right moves and $j$ left moves in the first $3j \leq 3n$ steps. Next we use Stirling's approximation to estimate ${q_j}^2$.

**Lemma 4.1.3** (Stirling's formula)**.** *For $m > 0$,*

$$\sqrt{2\pi m}\left(\frac{m}{e}\right)^m \leq m! \leq 2\sqrt{2\pi m}\left(\frac{m}{e}\right)^m .$$

Using this bound we have

$$\begin{aligned}
\binom{3j}{j} &= \frac{(3j)!}{j!(2j)!} \\
&\geq \frac{\sqrt{2\pi(3j)}\left(\frac{3j}{e}\right)^{3j}}{4\sqrt{2\pi j}\left(\frac{j}{e}\right)^{j}\sqrt{2\pi(2j)}\left(\frac{2j}{e}\right)^{2j}} \\
&= \frac{\sqrt{3}}{8\sqrt{\pi j}}\left(\frac{27}{4}\right)^{j} \\
&= \frac{c}{\sqrt{j}}\left(\frac{27}{4}\right)^{j} \qquad \text{where } c = \frac{\sqrt{3}}{8\sqrt{\pi}}
\end{aligned}$$

Thus, when $j > 0$,

$$\begin{aligned}
q_j &\geq \binom{3j}{j}\left(\frac{2}{3}\right)^{j}\left(\frac{1}{3}\right)^{2j} \\
&\geq \frac{c}{\sqrt{j}}\left(\frac{27}{4}\right)^{j}\left(\frac{2}{3}\right)^{j}\left(\frac{1}{3}\right)^{2j} \\
&\geq \frac{c}{\sqrt{j}}\frac{1}{2^j} .
\end{aligned}$$

This essentially tell us that the probability of success in a given phase drops no more than exponentially with the distance $j$ between the initial random assignment of the phase and the satisfying truth assignment. But by the Chernoff bounds, the probability that this distance is away from $n/2$, also drops exponentially. The tradeoff gives the running time.

More precisely, having established a lower bound for $q_j$ we now obtain a lower bound for $q$, the probability to reach state $n$ within $3n$ steps in case the initial state of the random walk

[2]This is one of the few cases in which we need to use an almost tight bound, e.g, Stirling's formula. A weaker bound such as $\binom{a}{b} \geq (a/b)^b$, although easier to apply, will not give the same bound on the running time. In fact, it will give running time $O(n(18/11)^n)$ (Why? Redo the calculations below with this bound).

is determined by a random assignment. To this end we have

$$\begin{aligned} q &\geq \sum_{j=0}^{n} \mathbb{P}(\text{walk starts at } n-j) \cdot q_j \\ &\geq \frac{1}{2^n} + \sum_{j=1}^{n} \binom{n}{j} \left(\frac{1}{2}\right)^n \frac{c}{\sqrt{j}} \frac{1}{2^j} \\ &\geq \frac{c}{\sqrt{n}} \left(\frac{1}{2}\right)^n \sum_{j=0}^{n} \binom{n}{j} \left(\frac{1}{2}\right)^j \\ &= \frac{c}{\sqrt{n}} \left(\frac{1}{2}\right)^n \left(\frac{3}{2}\right)^n \\ &= \frac{c}{\sqrt{n}} \left(\frac{3}{4}\right)^n . \end{aligned}$$

The number of phases to find a satisfying assignment is a geometric random variable with parameter $q$, so the expected number of phases to find a satisfying assignment is $1/q$. Thus the algorithm finds a satisfying assignment with probability at least $1/2$ after $N = 2/q$ phases. □

**Sources**

- Mitzenmacher and Upfal, 2nd edition, Section 7.1
- Lap Chi Lau, L06.pdf (page 1)

## 4.2 Markov chains and random walks

Consider a Markov chain $X$ with state space $\Omega$ and transition matrix $P$.

**Definition 4.2.1.** $X$ is *irreducible* if, for all $i \in \Omega$ and $j \in \Omega$, there is a $t \geq 0$ such that $P^t_{i,j} > 0$.

If $\Omega$ is finite, this is equivalent to saying that the graph representation of $X$ is strongly connected.

**Definition 4.2.2.** Given $i \in \Omega$ and $j \in \Omega$, the *hitting time* $H_{i,j}$ is a random variable giving the number of steps taken to visit state $j$ starting from state $i$. That is,

$$H_{i,j} = \min\{t > 0 : X_t = j \mid X_0 = i\} .$$

We define $r^t_{i,j} = \mathbb{P}(H_{i,j} = t)$ and $h_{i,j} = \mathbb{E}[H_{i,j}]$. A careful reader may note that our textbook defines the range of a random variable as $\mathbb{R}$ but the range of $H_{i,j}$ is $\mathbb{Z}_{\geq 1} \cup \{+\infty\}$. It is OK to include $+\infty$ though $h_{i,j}$ will be finite only if $\mathbb{P}(H_{i,j} = \infty) = 0$ (equivalently, only if $\sum_{t \geq 1} r^t_{i,j} = 1$).

**Definition 4.2.3.** A state $i \in \Omega$ is *recurrent* if $\sum_{t\geq 1} r_{i,i}^t = 1$. Otherwise, it is *transient*. $X$ is recurrent if all of its states are recurrent.

For example, consider a one-dimensional random walk which starts at 0, and moves to the right at each step with probability $7/8$ and to the left with probability $1/8$. The probability of being at the origin at step $2t$ is $\binom{2t}{t}(7/8)^t(1/8)^t < 2^{2t}(7/8)^t(1/8)^t = (7/16)^t$. Also, $\sum_{i=1}^{\infty}(7/16)^t = \frac{1}{1-\frac{7}{16}} - 1 = 7/9 < 1$. Thus, the origin is not recurrent (and neither are the other states).

On the other hand, if $\Omega = \{1,2\}$ and, for some $p \in (0,1)$, $P = \begin{pmatrix} p & 1-p \\ p & 1-p \end{pmatrix}$ then $\sum_{t\geq 1} r_{1,1}^t = \sum_{t\geq 1} p(1-p)^{t-1} = p\sum_{t\geq 0}(1-p)^t = p\frac{1}{1-(1-p)} = 1$, so state 1 is recurrent (and so is state 2).

If $i$ is recurrent, then $X$ will visit $i$ infinitely often. Otherwise, starting from $i$, it returns to $i$ with probability $q = \sum_{t\geq 1} r_{i,i}^t < 1$ so the number of visits that it makes to $i$ is a geometric random variable.

**Definition 4.2.4.** A recurrent state $i \in \Omega$ is *positive recurrent* if $h_{i,i} < \infty$. Otherwise, it is *null recurrent*.

In our two-state example, state 1 is positive recurrent because $h_{1,1}$ is a geometric random variable with parameter $p$, so $h_{1,1} = 1/p$.

Here is an example of a chain with null recurrent states. Let $\Omega$ be the positive integers and let $P$ be the transition matrix with $P_{i,i+1} = i/(i+1)$ and $P_{i,1} = 1/(i+1)$. Starting at state 1, the probability not to have returned to state 1 within $t$ steps is

$$\prod_{j=1}^{t} \frac{j}{j+1} = \frac{1}{t+1}\,.$$

Thus, $r_{1,1}^t = \frac{1}{t(t+1)} = \frac{1}{t} - \frac{1}{t+1}$. State 1 is recurrent, since $\sum_{t\geq 1}\frac{1}{t(t+1)} = 1$. (To see that the partial sums converge to 1, note that $\sum_{t=1}^{N}\frac{1}{t(t+1)} = \sum_{t=1}^{N}\frac{1}{t} - \sum_{t=1}^{N}\frac{1}{t+1} = 1 - \frac{1}{N+1}$.) Furthermore, $h_{1,1} = \sum_{t\geq 1} t\, r_{1,1}^t = \sum_{t\geq 1}\frac{1}{t+1} = \infty$, so state 1 is null recurrent.

Fortunately, finite Markov chains do not have null recurrent states. We will use the following theorem.

**Theorem 4.2.5.** *For any pair of states $i, j$ in a finite irreducible Markov chain, $h_{i,j} < \infty$.*

*Proof.* Let $d$ be the diameter of the underlying graph, i.e., the maximum distance between any two states, and let $\varepsilon > 0$ be the smallest positive entry of the transition matrix $P$. Then for any pair of states $i$ and $j$, if the chain is started in $i$ then it visits $j$ within $d$ steps with probability at least $\varepsilon^d$. We deduce that

$$\begin{aligned} h_{i,j} &= \mathbb{E}[H_{i,j}] \\ &= \sum_{t=1}^{\infty}\mathbb{P}(H_{i,j} \geq t) \\ &\leq \sum_{t=1}^{\infty}(1-\varepsilon^d)^{\lfloor (t-1)/d\rfloor} \\ &< \infty\,. \end{aligned}$$

□

**Definition 4.2.6.** The *period* of a state $i$ in a Markov chain is defined to be $\gcd\{m > 0 : P^m_{i,i} > 0\}$. A state is *aperiodic* if it has period 1. A Markov chain is aperiodic if all states are aperiodic.

A symmetric one-dimensional random walk is periodic, because it can only return to the origin in an even number of steps.

**Definition 4.2.7.** A state is *ergodic* if it is aperiodic and positive recurrent. A Markov chain is ergodic if all of its states are

The following is a corollary of Theorem 4.2.5

**Corollary 4.2.8.** *If a finite Markov chain is irreducible and aperiodic then it is ergodic.*

(By Theorem 4.2.5, every state is positive recurrent. Then the corollary follows from the definition of ergodic.)

### 4.2.1 Stationary Distributions

**Definition 4.2.9.** Let $X$ be a Markov chain with transition matrix $P$. A probability distribution $\pi$ on the set of states is called a *stationary distribution* of $X$ if

$$\pi = \pi P.$$

If a chain reaches a stationary distribution then it stays there. Note that if $P$ is the identity matrix, then every distribution is a stationary distribution.

Mitzenmacher and Upful refer to the following theorem as the "fundamental theorem of Markov chains".

**Theorem 4.2.10.** *Let $X$ be a finite, irreducible, aperiodic Markov chain with state space $\Omega$ and transition matrix $P$. Then*

1. *The chain has a unique stationary distribution $\pi$.*
2. *For all $i, j \in \Omega$, $\lim_{t\to\infty} P^t_{j,i}$ exists and is independent of $j$,*
3. *$\pi_i = \lim_{t\to\infty} P^t_{j,i} = 1/h_{i,i}$.*

Theorem 4.2.10 tells us that the stationary distribution represents a *time average*. On average, the chain visits state $j$ every $h_{j,j}$ steps, and thus spends proportion of time $\pi_j = 1/h_{j,j}$ in state $j$.

The theorem also tells us that the stationary distribution is the limiting distribution in the sense that the distribution of $X_t$ converges to $\pi$ as $t$ tends to infinity.

The condition that $X$ be aperiodic is not necessary for the existence of the stationary distribution, but it is necessary for the convergence. For example, if the transition matrix is $P = \left(\begin{smallmatrix} 0 & 1 \\ 1 & 0 \end{smallmatrix}\right)$, then the stationary distribution is $(1/2, 1/2)$, but the chain does not converge to this distribution from the starting distribution $(1, 0)$. Each state is visited every other time step.

One way to compute the stationary distribution is to solve the linear system $\pi = \pi P$ together with the condition $\sum_i \pi_i = 1$ Another way is called time-reversibility.

**Definition 4.2.11.** Let $X$ be a Markov chain with finite state space $\Omega$ and transition matrix $P$. $X$ is said to be *time-reversible* with respect to a distribution $\pi$ on its states if it satisfies the following equations (called the "detailed balance" equations): For all $i, j \in \Omega$, $\pi_i P_{i,j} = \pi_j P_{j,i}$.

**Theorem 4.2.12.** *Let $X$ be a Markov chain with finite state space $\Omega$ and transition matrix $P$. If $X$ is time-reversible with respect to a distribution $\pi$ on its states then $\pi$ is a stationary distribution of $X$.*

*Proof.* Consider the $j$'th entry of $\pi P$. We wish to show that this is $\pi_j$. The $j$'th entry of $\pi P$ is $\sum_{i\in\Omega} \pi_i P_{i,j}$. By detailed balance, this is $\sum_{i\in\Omega} \pi_j P_{j,i} = \pi_j \sum_{i\in\Omega} P_{j,i} = \pi_j$. □

If we already know (from Theorem 4.2.10) that $X$ has a unique stationary distribution, then of course, the stationary distribution from Theorem 4.2.12 is unique.

### 4.2.2 Random walks on undirected graphs

Let $G = (V, E)$ be a finite, undirected and connected graph. The *degree* of a vertex $v \in V$ is the number $d(v)$ of edges incident to $v$. A *random walk* on $G$ is a Markov chain with state space $V$. From a state $v$, the chain moves to a new vertex by following any edge adjacent to $v$, choosing which one to follow u.a.r.. For each edge $(u, v) \in E$ the probability of transitioning from $u$ to $v$ is $1/d(u)$.

**Claim 4.2.13.** A random walk on $G$ is aperiodic if and only if $G$ is not bipartite.

Clearly if a connected graph with at least two vertices is bipartite, the period of every state is 2. Conversely, if the graph is not bipartite it must contain an odd simple cycle. Then, by connectivity, every vertex $v$ is in an odd (not-necessarily simple) cycle. Since $v$ trivially belongs to a 2-cycle, because every edge creates a 2-cycle, $v$ has period 1.

**Theorem 4.2.14.** *A random walk on a connected graph $G$ that is not bipartite converges to a unique stationary distribution $\pi$, where*

$$\pi_v = \frac{d(v)}{2|E|}\,.$$

*Proof.* First we observe that $\pi$ is indeed a probability distribution. This is because $\sum_{v\in V} d(v) = 2|E|$ and thus

$$\sum_{v\in V} \pi_v = \sum_{v\in V} \frac{d(v)}{2|E|} = 1\,.$$

Next we establish detailed balance. This is easy because $\pi_v \frac{1}{d(v)} = 1/(2|E|)$ which is independent of $v$.

So we've established that $\pi$ is a stationary distribution. The fact that this stationary distribution is unique (and that the random walk converges to it) follows from Theorem 4.2.10, using the fact that the graph is not bipartite (so the chain is not periodic). □

**Corollary 4.2.15.** *Consider a random walk on a connected graph that is not bipartite. Let $h_{u,v}$ denote the expected number of steps to go from vertex $u$ to vertex $v$. Then for any vertex $u$ we have*

$$h_{u,u} = \frac{2|E|}{d(u)}.$$

**Theorem 4.2.16.** *Consider a random walk on a connected graph that is not bipartite. If $u$ and $v$ are adjacent then $h_{v,u} < 2|E|$.*

*Proof.* We give two different expressions for $h_{u,u}$:

$$\frac{2|E|}{d(u)} = h_{u,u} = \frac{1}{d(u)} \sum_{v \in N(u)} (1 + h_{v,u}).$$

The left-hand equation is from the above corollary and the right-hand equation comes from the system of linear equations defining $h_{i,j}$. It follows that

$$2|E| = \sum_{v \in N(u)} (1 + h_{v,u})$$

and thus $h_{v,u} < 2|E|$. □

The *cover time* of a graph $G$ is the maximum, over all vertices $v$, of the expected time that a random walk starting at $v$ takes to visit all other vertices.

**Theorem 4.2.17.** *The cover time of a non-bipartite connected graph $G$ with $n$ vertices and $m$ edges is at most $4nm$.*

*Proof.* Pick a spanning tree of $G$ and consider the cycle generated from a depth-first search of the spanning tree (in which all edges are taken twice). Let $v_0, \ldots, v_{2n-2} = v_0$ be the vertices in the cycle. Then the cover time is bounded by

$$\sum_{i=0}^{2n-3} h_{v_i,v_{i+1}} \leq (2n-2)2m \leq 4nm.$$

□

### 4.2.3 s-t connectivity

We can use the bound on the cover time of random walks to design an impressively simple space-efficient algorithm for the fundamental problem of $s-t$ connectivity. In this problem we are given an undirected graph with $n$ vertices, not necessarily connected. We are also given two vertices $s$ and $t$ and we want to find out whether $s$ and $t$ are connected.

*$s-t$ connectivity ($G$, $s$, $t$)*

- *start a random walk from $s$*
- *If $t$ is reached in $4n^3$ steps then return* reachable *else return* unreachable

Note that there are no false positives in the algorithm above.

Suppose for convenience that no component of $G$ is bipartite. (Otherwise, this can be dealt with, for example, by adding triangles).

If there is a path between $s$ and $t$ then the expected time for the random walk to go from $s$ to $t$ is at most the cover time $4nm \leq 2n^3$. By Markov's inequality, the algorithm outputs a path from $s$ to $t$ with probability at least $1/2$.

The above algorithm only needs to store a constant number of vertex-addresses at any one time. It follows that we can decide reachability in undirected graphs in $O(\log n)$ space using randomisation. Reingold showed the remarkable result that there is a deterministic algorithm that decides reachability in space $O(\log n)$.

**Sources**

- Mitzenmacher and Upfal, 2nd edition, Section 7.2-7.4
- Lap Chi Lau, L06.pdf (page 5)

## 4.3 Monte Carlo, Counting and Approximate Counting

### 4.3.1 The Monte Carlo method

The term *Monte Carlo method* refers to algorithms that estimate a value based on sampling.

**Example (Area of a region):** Suppose that we want to find the area of a 2-dimensional region $B$ that lies inside the unit square. We have access to an oracle that given a point $x$ returns whether $x \in B$.

The following is a very natural idea:

- *X=0*
- *Repeat $m$ times:*
  - *Select a random point $x$ in the unit square*
  - *If $x \in B$, then $X = X + 1$*
- *Return $X/m$*

Let $W$ be the output of the algorithm and let $W^*$ be the actual area of region $B$. Note that $\mathbb{E}[W] = W^*$. Also, by a Chernoff bound (Corollary 3.2.2), for $\varepsilon \in (0, 1)$ we have

$$\begin{aligned}\mathbb{P}(|W - W^*| \geq \varepsilon W^*) &= \mathbb{P}(m|W - W^*| \geq \varepsilon m W^*) \\ &\leq 2e^{\frac{-\varepsilon^2 m W^*}{3}}\end{aligned}$$

If we select $m = \Omega(\frac{1}{\varepsilon^2 W^*})$, this bound on the probability becomes a small constant.

Note however, that the required number of steps is inversely proportional to the area of the region. If $B$ is really small, we need many steps. Note that we cannot do much better if we only have access to an oracle and we don't know anything else about the region; we need $\Omega(1/W^*)$ trials even to find a single point in $B$.

We can generalise the above to get

**Theorem 4.3.1.** *Let $X_1, \ldots, X_m$ be i.i.d. indicator variables with $\mu = \mathbb{E}[X_i]$. If $m \geq \frac{3 \log \frac{2}{\delta}}{\varepsilon^2 \mu}$ and $\varepsilon \in (0,1)$, then*

$$\mathbb{P}\left(\left|\frac{1}{m}\sum_{i=1}^{m} X_i - \mu\right| \geq \varepsilon\mu\right) \leq \delta.$$

When we apply the Monte Carlo method, we typically have a function $f$ (whose range consists of non-negative real numbers) and we are interested in having an algorithm that takes an input instance $I$ and returns a good approximation to the desired value, $f(I)$. For example, the input $I$ might be a graph, and the function $f(I)$ might be the number of independent sets in $I$.

In order to describe what constitutes a good approximation algorithm, we need some definitions.

**Definition 4.3.2.** A *randomised approximation scheme* (RAS) for $f$ is a randomised algorithm $A$ that takes as input an instance $I$ and a rational error tolerance $\varepsilon \in (0,1)$ such that, for every $I$ and $\varepsilon$, the output $A(I,\varepsilon)$ satisfies

$$\mathbb{P}\left(|A(I,\varepsilon) - f(I)| \leq \varepsilon f(I)\right) \geq 3/4$$

.

**Definition 4.3.3.** A RAS for $f$ is said to be a *full polynomial randomised approximation scheme* (FPRAS) if its running time is bounded from above by a polynomial in $|I|$ (the size of the instance $I$) and $1/\varepsilon$.

The following theorem shows that the "3/4" in the definition of FPRAS can be arbitrarily strengthened. The same reasoning shows that there is nothing very special about the value "3/4" in the definition. Any number strictly between $1/2$ and $1$ would do.

**Theorem 4.3.4.** *If there is an FPRAS for $f$ then there is a randomised algorithm $A$ that takes as input an instance $I$, a rational error tolerance $\varepsilon \in (0,1)$ and a failure probability $\delta \in (0,1)$ such that, for every $I$ and $\varepsilon$, the output $A(I,\varepsilon,\delta)$ satisfies*

$$\mathbb{P}\left(|A(I,\varepsilon,\delta) - f(I)| \leq \varepsilon f(I)\right) \geq 1 - \delta. \tag{4.1}$$

*The running time of $A(I,\varepsilon,\delta)$ is bounded by a polynomial in $|I|$, $1/\varepsilon$, and $\log(1/\delta)$.*

*Proof.* Consider an input $I$, along with an error tolerance $\varepsilon$ and a failure probability $\delta$. Let $m = 24\lceil \log(1/\delta) \rceil$. Let $z_1, \ldots, z_m$ be outputs of the FPRAS with inputs $I$ and $\varepsilon$. Let $z$ be the median of $z_1, \ldots, z_m$. If $|z - f(I)| > \varepsilon f(I)$ then at most $m/2$ of the $z_i$'s satisfy $|z_i - f(I)| \leq \varepsilon f(I)$. Now we can use a Chernoff bound (Corollary 3.2.2) to bound the probability of this event. Let $X_i$ be the indicator random variable for the event that $|z_i - f(I)| \leq \varepsilon f(I)$. Let $p \geq 3/4$ be the probability that $X_i = 1$ (otherwise, $X_i = 0$). Let $X = \sum_i X_i$ with $\mathbb{E}[X] = mp$. By a Chernoff bound,

$$\begin{aligned}\mathbb{P}(X \leq m/2) = \mathbb{P}(X \leq (1 - (1 - 1/(2p)))mp) &\leq \exp(-mp(1 - 1/(2p))^2/2) \\ &\leq \exp(-m/24) \leq \delta,\end{aligned}$$

where the second to last inequality follows by noting that $p(1 - 1/(2p))^2$ is increasing for $p \in [3/4, 1)$ and is equal to $1/12$ for $p = 3/4$. □

A randomised algorithm achieving the guarantee in Equation (4.1) is said to be an $(\varepsilon, \delta)$ approximation for $f$.

### 4.3.2 The complexity class #P

Many of the problems for which we people design FPRASes come from a complexity class called #P. (This is pronounced as "number-P" or, sometimes, in America, as "sharp-P".) In order to define complexity classes, we have to encode instances over a finite alphabet. So fix a finite alphabet $\Sigma$ (for example, $\Sigma = \{0, 1\}$), and encode instances as elements of $\Sigma^*$ (strings of symbols from $\Sigma$). A decision problem can then be thought of as a predicate $\rho : \Sigma^* \to \{0, 1\}$.

For example, in Section 4.1, we studied 3-SAT, the problem of determining whether a formula in 3-CNF is satisfiable (whether there is a truth assignment to the variables that satisfies all clauses). The formula can be represented as a string in $\Sigma^*$, and an algorithm for 3-SAT would take such a string as input, and output 0 or 1, depending on whether the formula is satisfiable. We denote the length of a string $x \in \Sigma^*$ by $|x|$.

You have probably seen the complexity class NP defined in terms of Turing machines, but it is equivalent to define it as follows. The decision problem associated with the predicate $\rho$ is in NP if there exists a polynomial $p$ and a polynomial-time checkable predicate $\psi : \Sigma^* \times \Sigma^* \to \{0, 1\}$ such that $\rho(x) = 1$ if and only if there is a string $w \in \Sigma^*$ with $|w| \leq p(|x|)$ such that $\psi(x, w) = 1$. The predicate $\psi$ is sometimes called a "witness-checking predicate" because the string $w$ is a "witness" that $\rho(x) = 1$, and this is checked by checking whether $\psi(x, w) = 1$.

That is a lot to parse, but here is an example. Let $\rho$ be the predicate associated with 3-SAT. Let $\psi$ be a predicate that takes a string $x$ (representing a 3-CNF formula $\varphi$) and another string $w$ (representing a truth assignment to the variables of $\varphi$) such that $\psi(x, w) = 1$ iff the truth assignment represented by $w$ satisfies $\varphi$. Clearly, determining whether $\psi(x, w) = 1$ can be done in polynomial time. We can encode truth-assignments as bit strings, so that, if the formula $\varphi$ has $n$ variables, then $|w| \leq n$. Furthermore, $\rho(x) = 1$ iff there exists a truth assignment corresponding to some string $w$ such that $\psi(x, w) = 1$. Hence, 3-SAT is in NP.

Finally, we can define the class #P. A *counting problem* can be viewed as a function $f \colon \Sigma^* \to \mathbb{Z}_{\geq 0}$, mapping strings (representing instances) to natural numbers. Re-visiting our earlier example, the instance $I$ might be a string representing a graph, and the function $f(I)$ might be the number of independent sets in $I$.

A counting problem $f \colon \Sigma^* \to \mathbb{Z}_{\geq 0}$ is in FP (function-polynomial-time) if it can be computed in polynomial time. It is in the complexity class #P if there is a polynomial $p$ and a polynomial-time checkable predicate $\psi$ such that, for all $x \in \Sigma^*$,

$$f(x) = |\{w \in \Sigma^* \mid |w| \leq p(x) \wedge \psi(x, w) = 1\}|.$$

Thus, counting the satisfying assignments of a 3-CNF formula is in #P. In fact, counting the satisfying assignments of a 3-CNF formula is complete for #P with respect to polynomial-time Turing reductions, so there is no polynomial-time algorithm for counting the satisfying assignments of a 3-CNF formula unless every problem in #P has a polynomial-time algorithm.

A study of #P-completeness is beyond the scope of this course, but we will see examples of #P-complete problems that have FPRASes, so approximating #P-complete problems can (apparently) be easier than solving them exactly. The next section gives an example.

### 4.3.3 DNF counting

Let $\varphi \equiv C_1 \vee \cdots \vee C_t$ be a propositional formula in **disjunctive normal form (DNF)** over the set of variables $x_1, \ldots, x_n$. For example,

$$\varphi \equiv (x_1 \wedge \overline{x}_2 \wedge x_3) \vee (\overline{x}_2 \wedge x_4) \vee (\overline{x}_3 \wedge \overline{x}_4)\,.$$

The DNF counting problem asks us to compute the number $\#\varphi$ of truth assignments that satisfy $\varphi$. It is trivial to determine whether there exists a satisfying assignment of $\varphi$ — it suffices to satisfy any clause. However, counting satisfying assignments is #P-complete. (This is because $\#\overline{\varphi} + \#\varphi = 2^n$, but $\overline{\varphi}$ is equivalent (by de Morgan's law) to a CNF formula, so we know from the previous section that computing $\#\overline{\varphi}$ is #P-complete.)

Despite the fact that it is #P-complete to exactly count the satisfying assignments of a DNF formula, we will now give an FPRAS for this problem.

#### The naive algorithm

We start with a naive algorithm inspired by our algorithm for estimating the area of a region in the unit square.

- $X = 0$
- *repeat* $m$ *times:*
  - *generate a random truth assignment* $x$
  - *if* $\varphi(x) =$ *true then* $X = X + 1$
- *return* $\frac{X}{m} 2^n$

By Theorem 4.3.1, this algorithm achieves an $(\varepsilon, \delta)$-approximation when

$$m \geq \frac{3 \log(2/\delta)}{\varepsilon^2 \mu},$$

where $\mu$ is the fraction of satisfying truth assignments. Therefore, if the formula has at least $2^n/p(n)$ satisfying truth assignments, for some polynomial $p(n)$, the number of steps is polynomial. But if the formula has relative few satisfying truth assignments, the running time can be exponential. This situation is very similar to the case of estimating the area of a small region.

In conclusion, the above algorithm is not an FPRAS. But we can get an FPRAS by changing appropriately the sample space. In fact, we will solve a more general problem.

#### The union-of-sets sampling algorithm

The *union of sets* problem is as follows. There are $t$ subsets $H_1, \ldots, H_t$ of some finite set $U$ and we want to estimate the size of their union $H = H_1 \cup \cdots \cup H_t$. We assume that, for each $i$,

- we know $|H_i|$
- we can sample uniformly from $H_i$, and

- we can check whether $x \in H_i$, given $x \in U$.

Specifically, we assume that all three of these can be computed in time polynomial in the "size of the input" which is $t + \log(|U|)$.

Note that the use of the inclusion-exclusion formula is not practical since it contains $2^t - 1$ terms. Similarly, the naive algorithm of sampling uniformly from $U$, assuming that this is possible, will not give an FPRAS, in the same way that the naive algorithm does not solve the DNF counting problem.

The key idea for the solution is to find a new sample space in which the target set $H$ is reasonably dense. Let

$$W = \{(i, x) : 1 \leq i \leq t \text{ and } x \in H_i\}$$
$$H' = \{(i, x) \in W : (j, x) \notin W \text{ for } j < i\}.$$

Notice that $|H'| = |H|$. Also we can calculate the size of $W$ since $|W| = |H_1| + \cdots + |H_t|$. So we can estimate $|H|$ by estimating $\mu = |H'|/|W|$. Notice that $|W| \leq t|H|$ and so $\mu \geq 1/t$. Therefore we need to take at most $m = 3\log(2/\delta)t/\varepsilon^2$ samples.

It remains to observe that we can sample uniformly from $W$: we simply pick $i \in \{1, \ldots, t\}$ with probability

$$\frac{|H_i|}{\sum_{j=1}^t |H_j|} = \frac{|H_i|}{|W|}$$

and then pick $x \in H_i$ u.a.r.

*union-of-sets sampling algorithm*

- $X = 0$
- *repeat* $m$ *times:*
  - *select* $i$ *with probability* $|H_i|/\sum_{j=1}^t |H_j|$
  - *select a random element from* $H_i$
  - *if it does not belong to any* $H_j$*, with* $j < i$*, then* $X = X + 1$
- *output* $\frac{X}{m}\sum_{j=1}^t |H_j|$

By Theorem 4.3.1, this algorithm achieves an $(\varepsilon, \delta)$-approximation when

$$m \geq \frac{3\log(2/\delta)}{\varepsilon^2 \mu},$$

so we have

**Theorem 4.3.5.** *The union-of-sets sampling algorithm is an FPRAS for the union of sets problem, as long as the number of samples* $m$ *satisfies* $m \geq 3t\log(8)/\varepsilon^2$ *and is at most a polynomial in* $n$ *and* $1/\varepsilon$.

It is straightforward to cast the DNF counting problem as a special case of the union-of-sets problem. Suppose that $\varphi$ has $n$ variables and $t$ clauses. The set $U$ consists of all $2^n$ truth

assignments of the variables of $\varphi$. For $1 \leq i \leq t$, $H_i$ is the set of truth assignments that satisfy the $i$'th clause, $C_i$. The "size of the input" for the union-of-sets problem is $t+\log(|U|) = t+n$, and this does correspond to the size of the input $\varphi$.

We can assume that each variable occurs at most once in each clause — repeated literals have no effect, and a clause is not satisfiable if it contains a literal and its negation, so in that case the clause can be removed. Then $|H_i| = 2^{n-l_i}$ where $l_i$ is the number of literals in $C_i$. This can be computed in $\text{poly}(n)$ time. Also it is straightforward to sample uniformly from $H_i$: the variables occurring in $C_i$ are determined and all variables not occurring in $C_i$ have their truth values chosen u.a.r. Finally, given a truth assignment $x \in U$, we can check in polynomial time whether $x \in H_i$.

**Sources**

- Mitzenmacher and Upfal, 2nd edition, Section 11.1 - 11.2
- Jerrum, Counting Sampling and Integrating: Algorithms and Complexity, Chapter 2

## 4.4 Sampling Algorithms and Markov Chain Monte Carlo

### 4.4.1 Sampling

We defined a counting problem $f : \Sigma^* \to \mathbb{Z}_{\geq 0}$ as the problem of computing, or approximately computing, a function $f$ from instances in $\Sigma^*$ to natural numbers. Similarly, we define a sampling problem by defining, for each instance $x \in \Sigma^*$ a set $\Omega(x)$ of objects (from which we want to sample) and a distribution $\mathcal{D}(x)$ on $\Omega(x)$ – the desired distribution. Often, $\mathcal{D}(x)$ is the uniform distribution on $\Omega(x)$.

For example, an instance $x \in \Sigma^*$ might encode a graph, $\Omega(x)$ might encode the set of independent sets of the graph, and $\mathcal{D}(x)$ might be the uniform distribution on $\Omega(x)$.

Given an input $x \in \Sigma^*$, the goal is to sample approximately from the distribution $\mathcal{D}(x)$. We next define distances between distributions so that we can define the notion of an approximate sampler.

**Definition 4.4.1.** The *total variation distance* between two distributions $\pi_1$ and $\pi_2$ on a finite set $S$ is given by

$$||\pi_1 - \pi_2||_{\text{TV}} = \frac{1}{2} \sum_{s \in S} |\pi_1(s) - \pi_2(s)| .$$

Sometimes the total variation distance is just referred to as the "variation distance". It is half of the $L_1$ distance between $\pi_1$ and $\pi_2$. The reason for the factor of $1/2$ in the definition is the following lemma, which gives an alternative characterisation of variation distance.

We use the following notation in the lemma, and elsewhere. Given a distribution $\pi$ on a set $S$ and any subset $T$ of $S$, we define $\pi(T)$ to be $\sum_{x \in T} \pi(x)$.

**Lemma 4.4.2.** *If $\pi_1$ and $\pi_2$ are distributions on a finite set $S$ then*

$$||\pi_1 - \pi_2||_{TV} = \max_{A \subseteq S} |\pi_1(A) - \pi_2(A)|.$$

*Proof.* Let $S^+ = \{x \in S : \pi_1(x) \geq \pi_2(x)\}$ and $S^- = \{x \in S : \pi_1(x) < \pi_2(x)\}$. From the definition of variation distance

$$\begin{aligned} ||\pi_1 - \pi_2||_{\text{TV}} &= \frac{1}{2}\sum_{x \in S^+}(\pi_1(x) - \pi_2(x)) + \frac{1}{2}\sum_{x \in S^-}(\pi_2(x) - \pi_1(x)) \\ &= \frac{1}{2}((\pi_1(S^+) - \pi_2(S^+)) + (\pi_2(S^-) - \pi_1(S^-))). \end{aligned}$$

But from $\pi_1(S^+) + \pi_1(S^-) = \pi_2(S^+) + \pi_2(S^-) = 1$ we have $\pi_1(S^+) - \pi_2(S^+) = \pi_2(S^-) - \pi_1(S^-)$. Thus we have

$$||\pi_1 - \pi_2||_{\text{TV}} = \pi_1(S^+) - \pi_2(S^+) = \pi_2(S^-) - \pi_1(S^-).$$

But the set $A \subseteq S$ that maximises $\pi_1(A) - \pi_2(A)$ is $A = S^+$. Similarly, the set $A \subseteq S$ that maximises $\pi_2(A) - \pi_1(A)$ is $A = S^-$. □

To illustrate the appropriateness of total variation distance, suppose that we perfectly shuffle a deck of cards, except that we leave the bottom card—say the ace of spades—unchanged. Let the resulting distribution on permutations of the set of cards be $\pi_1$. Let $\pi_2$ be the uniform distribution. Then we have $||\pi_1 - \pi_2||_{\text{TV}} \geq 51/52$ since, if $A$ is the event that the bottom card is the ace of spades, then $\pi_1(A) = 1$ and $\pi_2(A) = 1/52$.

Using the notion of variation distance, we can now define what it means to be a good approximate sampling algorithm.

**Definition 4.4.3.** Consider a sampling problem which defines a distribution $\mathcal{D}(x)$ on $\Omega(x)$ for each input $x \in \Sigma^*$. An $\varepsilon$-approximate sampler for this problem is an algorithm $A$ that takes an input $x \in \Sigma^*$ and returns an output from a distribution $A(x)$ such that $||A(x) - \mathcal{D}(x)||_{\text{TV}}| \leq \varepsilon$.

**Definition 4.4.4.** Consider a sampling problem which associates the uniform distribution $\mathcal{D}(x)$ on the set $\Omega(x)$ with each input $x \in \Sigma^*$. An $\varepsilon$-approximate sampler $A$ for this problem is said to be a *polynomial almost uniform sampler (PAUS)* for $\Omega$ if its run-time is at most a polynomial in $|x|$ and $1/\varepsilon$. It is said to be a *fully polynomial almost uniform sampler (FPAUS)* for $\Omega$ if its run-time is at most a polynomial in $|x|$ and $\log(1/\varepsilon)$.

Note the difference between the definitions of FPRAS and FPAUS. The running time of an FPRAS is polynomial in $1/\varepsilon$. This is the right notion of approximability because doing better than this is computationally difficult. On the other hand, the running time of an FPAUS has to be even smaller — it has to be at most a polynonmial in $\log(1/\varepsilon)$. This turns out to be feasible, so again it is the "right" definition. This difference is not important for "self-reducible" problems, which we will study below. For these problems a PAUS can be "powered up" to obtain an FPAUS.

### 4.4.2 From sampling to counting

Consider the independent set example where $\Omega(G)$ denotes the set of independent sets of a graph $G$ and $\mathcal{D}(G)$ is the uniform distribution on $\Omega(G)$. The associated counting and sampling problems are "self-reducible" which, informally, means that you can express $\Omega(G)$ in terms of sets $\Omega(G')$ for smaller graphs $G'$. For a self-reducible problem, a PAUS can be turned into an FPRAS. We will demonstrate this for the independent set problem.

**Theorem 4.4.5.** *Given a PAUS for sampling the independent sets of an input graph, we can construct an FPRAS for approximately counting independent sets.*

*Proof.* A naive approach for approximately counting independent sets is to sample uniform subsets of vertices, and check how many are independent sets. This fails for the same reason that the naive approach to the DNF counting problem fails: the proportion of independent sets among arbitrary sets could be exponentially small. Instead we pursue a more subtle approach, exploiting the *self-reducible* structure of the problem.

Consider an input graph $G = (V, E)$, where $E = \{e_1, e_2, \ldots, e_m\}$ and $|V| = n$. For each $i \in \{0, \ldots, m\}$ define the subgraph $G_i$ of $G$ by $G_i = (V, \{e_1, \ldots, e_i\})$. Thus $G_0$ has no edges and $G_m$ is $G$ itself. The key idea is that the number of independent sets in $G$ can be written as

$$|\Omega(G)| = |\Omega(G_m)| = \frac{|\Omega(G_m)|}{|\Omega(G_{m-1})|} \frac{|\Omega(G_{m-1})|}{|\Omega(G_{m-2})|} \cdots \frac{|\Omega(G_1)|}{|\Omega(G_0)|} |\Omega(G_0)| .$$

Since $G_0$ has no edges, $|\Omega(G_0)| = 2^n$. Considering the successive ratios

$$\varrho_i = \frac{|\Omega(G_i)|}{|\Omega(G_{i-1})|} ,$$

we have $|\Omega(G)| = 2^n \prod_{i=1}^m \varrho_i$. Thus we can approximate the number $|\Omega(G)|$ of independent sets by approximating each $\varrho_i$.

The theorem will follow from the following two claims.

**Claim 1.** To get an $(\varepsilon, 1/4)$-approximation for $|\Omega(G)|$, it suffices to get an $(\varepsilon/(2m), 1/(4m))$-approximation for each $\varrho_i$.

**Proof:** Suppose we have an $(\varepsilon/(2m), 1/(4m))$-approximation for each $\varrho_i$. Using a union bound, with probability at least $1 - m(1/(4m)) \geq 3/4$, the computed estimates $\hat{\varrho}_i$ each satisfy $|\hat{\varrho}_i - \varrho_i| \leq \varepsilon \varrho_i/(2m)$. Thus,

$$1 - \varepsilon \leq e^{-\varepsilon} \leq \left(1 - \frac{\varepsilon}{2m}\right)^m \leq \prod_{i=1}^m \frac{\hat{\varrho}_i}{\varrho_i} \leq \left(1 + \frac{\varepsilon}{2m}\right)^m \leq e^{\varepsilon/2} \leq 1 + \varepsilon.$$

Here we used $1 + x \leq e^x$, which we've already seen, for all $x$ — we use $x = \varepsilon/(2m)$. We also used $1 - x/2 \geq e^{-x}$, which holds for $x \in [0, 1]$ — we use $x = \varepsilon/(m)$. Finally, we use $e^{\varepsilon/2} \leq 1 + \varepsilon$ for $\varepsilon \in (0, 1)$.

**Claim 2.** Using the PAUS, we can get an $(\varepsilon/(2m), 1/(4m))$-approximation for $\varrho_i$ in time which is at most a polynomial in $n$ and $1/\varepsilon$.

**Proof:**

We want to establish an $(\varepsilon/(2m), 1/(4m))$-approximation for $\varrho_i = |\Omega(G_i)|/|\Omega(G_{i-1})|$.

The first step is to show that $3/4 \leq \varrho_i \leq 1$. The upper bound is easy, since every independent set of $G_i$ is an independent set of $G_{i-1} = G_i - e_i$. The other direction is also easy: Suppose that $e_i = (u, v)$ and partition the independent sets of $G_{i-1}$ into four sets — $\Omega_{u,v}$ for those that contain $u$ and $v$, $\Omega_{u,\overline{v}}$ for those that contain $u$ but not $v$, $\Omega_{\overline{u},v}$ for those that contain $v$ but not $u$, and $\Omega_{\overline{u},\overline{v}}$ for those that contain neither $u$ nor $v$. Clearly, $\Omega_{u,\overline{v}} \cup \Omega_{\overline{u},v} \cup \Omega_{\overline{u},\overline{v}}$ is contained in $\Omega(G_i)$. But each of these three sets is at least as big as $\Omega_{u,v}$ since there is a bijection from $\Omega_{u,v}$ into each of these three sets by adjusting whether $u$ or $v$ is in the independent set.

The algorithm for estimating $\varrho_i$ is then to draw an appropriate number, $N_i$, of samples from $\Omega(G_{i-1})$, using the PAUS, and calculate the fraction of them that are in $\Omega(G_i)$.

Let $\mu_i$ be the probability that the PAUS returns a sample in $\Omega(G_i)$. By running the PAUS with parameter $\varepsilon/(8m)$ the bound on the total variation distance gives $|\varrho_i - \mu_i| \leq \frac{\varepsilon}{8m}$. Using our bounds on $\varrho$, this gives $|\varrho_i - \mu_i| \leq \frac{\varepsilon}{6m}\varrho_i$.

Now fix $\delta = 1/(4m)$. By a Chernoff bound (Theorem 4.3.1), as long as we take $N_i \geq 3\log(2/\delta)/((\varepsilon/(8m))^2\mu_i)$ samples we get, with probability at least $1-\delta$, an estimate $\hat{\varrho}_i$ satisfying $|\hat{\varrho}_i - \mu_i| \leq \frac{\varepsilon}{8m}\mu_i \leq \frac{\varepsilon}{8m}(1+\frac{\varepsilon}{6m})\varrho_i \leq \frac{\varepsilon}{4m}\varrho_i$.

Putting these together, with probability at least $1-\delta$, $|\hat{\varrho}_i - \varrho_i| \leq \frac{\varepsilon}{2m}\varrho_i$, as required.

Note that the number of samples required is only $\lceil 3\log(2/\delta)/((\varepsilon/(8m))^2\mu_i)\rceil$. This meets the running time bound since $\mu_i \geq \varrho_i - \varepsilon/(8m) \geq 1/2$. □

### 4.4.3 Markov Chain Monte Carlo

One of the main uses of Markov chains is for sampling. This is based on two ideas: first, that we can realise a given distribution as the stationary distribution of a Markov chain, and second, that we can compute bounds on the rate of convergence of a Markov chain to its stationary distribution. In this section, we concentrate on the first part, getting the stationary distribution right.

We've already seen in Section 4.2.2 that a random walk on a connected graph $G = (V, E)$ that is not bipartite converges to a unique stationary distribution $\pi$, where $\pi_v = d(v)/(2|E|)$. If we'd like instead to have the stationary distribution be uniform on $V$ then we change the transition probabilities.

Let $\Delta$ be the maximum degree of any vertex in $V$. Consider the Markov chain $X$ with state space $\Omega = V$ and transition matrix $P$ defined as follows.

$$P_{x,y} = \begin{cases} 1/\Delta & \text{if } (x,y) \in E \\ 0 & \text{if } x \neq y \text{ and } (x,y) \notin E \\ 1 - d(x)/\Delta & \text{if } y = x \end{cases}$$

This chain is time-reversible with respect to the uniform distribution because $P_{x,y} = P_{y,x}$. Thus, the uniform distribution is a stationary distribution. If $X$ is irreducible and aperiodic then this is the unique stationary distribution. Since $G$ is a connected undirected graph $X$ is irreducible (it is possible to get from any vertex to any other one). Since $G$ is not bipartite, it is aperiodic.

If $G$ is bipartite, we can add self-loops to destroy the periodicity by defining $P$ as follows:

$$P_{x,y} = \begin{cases} 1/(2\Delta) & \text{if } (x,y) \in E \\ 0 & \text{if } x \neq y \text{ and } (x,y) \notin E \\ 1 - d(x)/(2\Delta) & \text{if } y = x \end{cases}$$

### 4.4.4 The Metropolis Algorithm

We now describe the *Metropolis algorithm*, which is a general way to add self-loop probabilities to construct a chain with a desired stationary distribution $\pi$.

.

**Theorem 4.4.6** (Metropolis algorithm)**.** *Suppose that $(\Omega, E)$ is a connected undirected graph with maximum degree $\Delta$. Let $\pi$ be a distribution on $\Omega$ with $\pi_x > 0$ for all $x \in \Omega$. Fixing $C > \Delta$, consider a Markov chain with state space $\Omega$ and transition matrix $P$, where*

$$P_{x,y} = \begin{cases} (1/C)\min(1, \pi_y/\pi_x) & \text{if } (x,y) \in E \\ 0 & \text{if } x \neq y \text{ and } (x,y) \notin E \\ 1 - \sum_{z:z\neq x} P_{x,z} & \text{if } y = x. \end{cases}$$

*Then $\pi$ is the unique stationary distribution of this chain.*

*Proof.* We first show that the chain satisfies detailed balance with respect to $\pi$. Consider $(x,y) \in E$. We will show that $\pi_x P_{x,y} = \pi_y P_{y,x}$. To do this, suppose without loss of generality that $\pi_x \leq \pi_y$ (otherwise, swap $x$ and $y$). Then $P_{x,y} = 1/C$ and $P_{y,x} = (1/C) \cdot (\pi_x/\pi_y)$. Thus $\pi_x P_{x,y} = \pi_x(1/C) = \pi_y P_{y,x}$, as required.

To show that $\pi$ is unique, note that the chain is irreducible (since the graph is connected) and aperiodic (since every node has a self-loop).

□

### 4.4.5 Example: The Hard-Core Gibbs Measure

Given a graph $G = (V, E)$, let $\mathcal{I}(G)$ denote the set of independent sets of $G$. We first give a Markov chain whose stationary distribution is the uniform distribution on $\mathcal{I}(G)$. The state space is $\Omega = \mathcal{I}(G)$. The Markov chain is a sequence $X_0, X_1, \ldots$ of states in $\Omega$ where the transition from $X_i$ to $X_{i+1}$ is given as follows.

- *Choose $v$ u.a.r. from $V$*
- $X' \leftarrow X_i \oplus \{v\}$
- *If $X' \in \mathcal{I}(G)$ then $X_{i+1} \leftarrow X'$*
- *Else $X_{i+1} \leftarrow X_i$*

The Markov chain is irreducible. To move between independent sets $I$ and $J$ first move from $I$ to the empty independent set, then put vertices back in to get $J$. As long as $G$ has an edge, the chain is aperiodic, since it can stay still from the independent set containing one endpoint (by choosing the other). Finally, it satisfies detailed balance with respect to the uniform distribution on $\Omega = \mathcal{I}(G)$. Given two distinct independent sets $I$ and $J$ the chain can only move directly between $I$ and $J$ if they differ in a single vertex. In that case, $P_{I,J} = P_{J,I} = 1/n$. Thus, the uniform distribution on $\Omega$ is the unique stationary distribution.

In the hard-core model from statistical physics, there is a parameter $\lambda > 0$. The quantity $Z(G) = \sum_{I \in \mathcal{I}(G)} \lambda^{|I|}$ is called the "partition function" and the Gibbs measure is the probability on $\mathcal{I}(G)$ that assigns probability $\pi_I = \lambda^{|I|}/Z(G)$ to each independent set $I \in \mathcal{I}(G)$. We can use the Metropolis algorithm to sample from the Gibbs measure. In particular, consider the Markov chain with state space $\Omega = \mathcal{I}(G)$ where the transition from $X_i$ to $X_{i+1}$ is given as follows.

- *Choose $v$ u.a.r. from $V$*
- $X' \leftarrow X_i \oplus \{v\}$

- *If* $X' \in \mathcal{I}(G)$
  - *With probability* $\min(1, \pi_{X'}/\pi_{X_i})$, $X_{i+1} \leftarrow X'$
  - *With the remaining probability* $X_{i+1} \leftarrow X_i$
- *Else* $X_{i+1} \leftarrow X_i$

Since $\lambda > 0$, every independent set $I$ has $\pi_I > 0$, so the arguments that the chain is irreducible and aperiodic are unchanged. But the new chain satisfies detailed balance with respect to $\pi$: Consider $I$ and $I'$ that differ in one vertex with $\pi_I \leq \pi_{I'}$. Then $P_{I,I'} = 1/n$ and $P_{I',I} = (1/n)(\pi_I/\pi_{I'})$ so $\pi_I P_{I,I'} = \pi_I(1/n) = \pi_{I'} P_{I',I}$.

To implement the algorithm we don't need to compute $\pi_{X'}$ and $\pi_{X_i}$. Computing these is difficult, because we need to normalise by computing $Z(G)$, which has exponentially many terms. We just need to compute the ratio $\pi_{X'}/\pi_{X_i}$. If $X'$ has one more vertex than $X_i$ then this is $\lambda$. Otherwise, $X'$ has one fewer vertex than $X_i$ and it is $1/\lambda$.

**Sources**

- Mitzenmacher and Upfal, 2nd edition, Section 11.3, 11.4
- Jerrum, Counting Sampling and Integrating: Algorithms and Complexity, Chapters 3.1,3.2

## 4.5 Mixing Times and Coupling

### 4.5.1 Mixing time

Let $(X_t)$ be a Markov chain with finite state space $\Omega$, transition matrix $P$ and a unique stationary distribution $\pi$. Recall that $P^t_{x,y}$ is the probability that $X_t = y$, given that $X_0 = x$. We use the notation $P^t_{x,\cdot}$ to denote the distribution of $X_t$, given that $X_0 = x$.

We define $d_{\mathrm{TV}x}(t) = ||P^t_{x,\cdot} - \pi||_{\mathrm{TV}}$ and $d_{\mathrm{TV}}(t) = \max_{x\in\Omega} d_{\mathrm{TV}x}(t)$. Given $\varepsilon \in (0,1)$, the *mixing time* from state $x$ is defined to be $\tau_x(\varepsilon) = \min\{t \mid d_{\mathrm{TV}x}(t) \leq \varepsilon\}$ and the mixing time of the chain to be $\tau(\varepsilon) = \max_{x\in\Omega} \tau_x(\varepsilon)$. The following lemma shows that $d_{\mathrm{TV}x}(t)$ is monotone decreasing, so that $d_{\mathrm{TV}x}(t) \leq \varepsilon$ for all $t \geq \tau_x(\varepsilon)$.

**Lemma 4.5.1.** *$d_{TVx}(t)$ is a non-increasing function of $t$.*

*Proof.* By the definition of total variation distance

$$d_{\mathrm{TV}x}(t+1) = \frac{1}{2}\sum_{y\in\Omega} |P^{t+1}_{x,y} - \pi(y)| = \frac{1}{2}\sum_{y\in\Omega}\left|\sum_{z\in\Omega} P^t_{x,z}P_{z,y} - \sum_{z\in\Omega}\pi(z)P_{z,y}\right|$$
$$= \frac{1}{2}\sum_{y\in\Omega}\left|\sum_{z\in\Omega} P_{z,y}\left(P^t_{x,z} - \pi(z)\right)\right|.$$

By the triangle inequality, this is at most

$$\frac{1}{2}\sum_{y\in\Omega}\sum_{z\in\Omega} P_{z,y}\left|P^t_{x,z} - \pi(z)\right| = \frac{1}{2}\sum_{z\in\Omega}\left|P^t_{x,z} - \pi(z)\right|\sum_{y\in\Omega} P_{z,y} = d_{\mathrm{TV}x}(t).$$

□

A Markov chain is said to be *rapidly mixing* if its mixing time $\tau(\varepsilon)$ is polynomial in $\log(1/\varepsilon)$ and the size of the problem. For example, we've seen a Markov chain for sampling the independent sets of a graph $G = (V, E)$. Such a chain would be rapidly mixing if its mixing time were polynomial in $\log(1/\varepsilon)$ and $n = |V|$.

One of the simplest methods for proving that a Markov chain is rapidly mixing is *coupling*.

**Definition 4.5.2.** A *coupling* of a Markov chain $(M_t)$ with finite state space $\Omega$ and transition matrix $P$ is a Markov chain $(Z_t) = (X_t, Y_t)$ with state space $\Omega \times \Omega$ such that:

$$\mathbb{P}(X_{t+1} = x' \mid X_t = x \wedge Y_t = y) = P_{x,x'}$$
$$\mathbb{P}(Y_{t+1} = y' \mid X_t = x \wedge Y_t = y) = P_{y,y'},$$

and $X_t = Y_t$ implies $X_{t+1} = Y_{t+1}$.

So the coupling contains two copies $(X_t)$ and $(Y_t)$ of the Markov chain $(M_t)$, but these do not evolve independently. Once they "couple", in the sense that $X_t = Y_t$, they evolve together, following the transition rules of $(M_t)$. Each copy $(X_t)$ and $(Y_t)$ is a faithful copy of $(M_t)$.

One obvious coupling is to consider two copies of $(M_t)$ operating independently until coupling, so that for distinct states $x$ and $y$

$$\mathbb{P}((X_{t+1}, Y_{t+1}) = (x', y') \mid (X_t, Y_t) = (x, y)) = P_{x,x'} P_{y,y'}$$

However this example is not useful for us. Instead, we correlate the evolution of $(X_t)$ and $(Y_t)$ so that these Markov chains are likely to couple quickly.

**Lemma 4.5.3** (Coupling lemma)**.** *Let $(X_t, Y_t)$ be a coupling of a Markov chain $(M_t)$ with finite state space $\Omega$, transition matrix $P$, and a unique stationary distribution $\pi$. Suppose that $t\colon [0,1] \to \mathbb{Z}_{\geq 0}$ is such that for all $x \in \Omega$, $y \in \Omega$ and $\varepsilon \in (0,1)$*

$$\mathbb{P}(X_{t(\varepsilon)} \neq Y_{t(\varepsilon)} \mid X_0 = x \wedge Y_0 = y) \leq \varepsilon.$$

*Then for all $x \in \Omega$ and $y \in \Omega$,*

$$||P^{t(\varepsilon)}_{x,\cdot} - P^{t(\varepsilon)}_{y,\cdot}||_{TV} \leq \varepsilon \tag{4.2}$$

*and the mixing time $\tau(\varepsilon)$ of $(M_t)$ is at most $t(\varepsilon)$.*

*Proof.* Fix any $x \in \Omega$, $y \in \Omega$ and $A \subseteq \Omega$. Then, by the definition of coupling (since $X_t$ is a faithful copy of $M_t$),

$$\begin{aligned}\mathbb{P}(M_{t(\varepsilon)} \in A \mid M_0 = x) &= \mathbb{P}(X_{t(\varepsilon)} \in A \mid X_0 = x \wedge Y_0 = y) \\ &\geq \mathbb{P}(X_{t(\varepsilon)} = Y_{t(\varepsilon)} \wedge Y_{t(\varepsilon)} \in A \mid X_0 = x \wedge Y_0 = y) \\ &= 1 - \mathbb{P}(X_{t(\varepsilon)} \neq Y_{t(\varepsilon)} \vee Y_{t(\varepsilon)} \notin A \mid X_0 = x \wedge Y_0 = y).\end{aligned}$$

By a union bound, this is at least

$$1 - (\mathbb{P}(X_{t(\varepsilon)} \neq Y_{t(\varepsilon)} \mid X_0 = x \wedge Y_0 = y) + \mathbb{P}(Y_{t(\varepsilon)} \notin A \mid X_0 = x \wedge Y_0 = y)),$$

which by the definition of coupling (since $Y_t$ is a faithful copy of $M_t$), is equal to

$$1 - (\mathbb{P}(X_{t(\varepsilon)} \neq Y_{t(\varepsilon)} \mid X_0 = x \wedge Y_0 = y) + \mathbb{P}(M_{t(\varepsilon)} \notin A \mid M_0 = y)).$$

Using the condition on $t$ in the lemma statement, this is at least $\mathbb{P}(M_{t(\varepsilon)} \in A \mid M_0 = y) - \varepsilon$. By symmetry, we also have

$$\mathbb{P}(M_{t(\varepsilon)} \in A \mid M_0 = y) \geq \mathbb{P}(M_{t(\varepsilon)} \in A \mid M_0 = x) - \varepsilon.$$

So using the definition of variation distance (Lemma 4.4.2), we get Equation (4.2). To get the bound on the mixing time from (4.2), just choose $y$ from the stationary distribution $\pi$ of $(M_t)$ so that $P^{t(\varepsilon)}_{y,\cdot} = \pi$. □

**Example: Card shuffling**

Consider a Markov chain for shuffling a deck of $n$ cards. At each step, a card is chosen independently and u.a.r. and put on the top of the deck.

Clearly, the state space is finite, and the chain is irreducible and aperiodic, so it has a unique stationary distribution. To see that this stationary distribution is uniform on the $n!$ permutations, consider the system $(\pi_1, \ldots, \pi_{n!})P = (\pi_1, \ldots, \pi_{n!})$. To see that the vector $(\pi_1, \ldots, \pi_{n!}) = (1/n!, \ldots, 1/n!)$ is a solution, Consider the value $\pi_i$ on the right-hand-side. On the left-hand-side, there are exactly $n$ states that could move to state $i$ (these are the $n$ states derived from state $i$ by moving the top card somewhere). Each of these has probability $1/n!$ on the left, and the probability of transitioning to $i$ is $1/n$. Thus, we obtain $\pi_i = 1/n!$, as desired.

Now let's consider the mixing time. The naive coupling based on selecting a random position $j$, the same in both decks, and moving the $j$'th card to the top does not work, because the chains will never couple (unless they start at the same state).

Here is a variant which brings the two copies together.

- Choose position $j$ u.a.r. from $\{1, \ldots, n\}$ and obtain $X_{t+1}$ from $X_t$ by moving the $j$'th card to the top. Call this card $C$.
- Obtain $Y_{t+1}$ from $Y_t$ by moving card $C$ to the top.

To see that it is a valid coupling, we have to show that $(Y_t)$ is a faithful copy of the original chain. So consider any position $k$ — let's calculate the probability that this position gets moved to the top by the transition from $Y_t$ to $Y_{t+1}$. Let $C'$ be the card in position $k$ in $Y_t$. Let $j$ be the position of $C'$ in $X_t$. The probability that $j$ is chosen is $1/n$, so this is the probability that position $k$ is chosen in the transition from $Y_t$ to $Y_{t+1}$.

Now let's see how long it takes to couple. Note that when a card $C$ is moved to the top of the deck in $(X_t)$ and $(Y_t)$ it thereafter occupies the same position in both decks. The probability that a given $C \in \{1, \ldots, n\}$ is chosen in each transition is $1/n$.

Thus, we have a coupon collector problem. The probability that a given card $C$ has not been chosen in $t$ steps is at most $(1-1/n)^t \leq \exp(-t/n)$ so, by a union bound, the probability that the chain has not coupled in $t$ steps is at most $n\exp(-t/n) = \exp(-t/n+\log n)$. Choosing $t = n(\log(1/\varepsilon) + \log(n))$ gives that this probability is at most $\varepsilon$. By the coupling lemma, the mixing time is at most $n(\log(1/\varepsilon) + \log(n))$

**Example: Token ring**

Consider a Markov chain representing a token moving round a ring. In each time step, the token stays where it is with probability $1/2$ and moves clockwise with probability $1/2$. The set of states is the set $\{0, 1, \ldots, n-1\}$ of positions of the token. Each state $i$ makes a transition to itself with probability $1/2$ and to $i \oplus 1$ with probability $1/2$, where $\oplus$ denotes addition modulo $n$.

Obviously, the chain is finite, irreducible and aperiodic, so it has a unique stationary distribution. To see that this stationary distribution is uniform, consider solving $(\pi_0, \ldots, \pi_{n-1})P = (\pi_0, \ldots, \pi_{n-1})$. For example, for $n = 4$,

$$(1/4, 1/4, 1/4, 1/4) \begin{pmatrix} 1/2 & 1/2 & 0 & 0 \\ 0 & 1/2 & 1/2 & 0 \\ 0 & 0 & 1/2 & 1/2 \\ 1/2 & 0 & 0 & 1/2 \end{pmatrix} = (1/4, 1/4, 1/4, 1/4).$$

We show that the mixing time $\tau(\varepsilon)$ is $O(n^2 \log(1/\varepsilon))$.

To this end, define a coupling $(X_t, Y_t)$ by having $(X_t)$ and $(Y_t)$ behave like independent copies of the Markov chain if $X_t \neq Y_t$, and otherwise have them behave identically. Formally, suppose $(X_t, Y_t) = (i, j)$. If $i = j$, then

$$\begin{aligned} \mathbb{P}((X_{t+1}, Y_{t+1}) = (i, j)) &= 1/2 \\ \mathbb{P}((X_{t+1}, Y_{t+1}) = (i \oplus 1, j \oplus 1)) &= 1/2 \end{aligned}$$

otherwise we define

$$\begin{aligned} \mathbb{P}((X_{t+1}, Y_{t+1}) = (i, j)) &= 1/4 \\ \mathbb{P}((X_{t+1}, Y_{t+1}) = (i, j \oplus 1)) &= 1/4 \\ \mathbb{P}((X_{t+1}, Y_{t+1}) = (i \oplus 1, j)) &= 1/4 \\ \mathbb{P}((X_{t+1}, Y_{t+1}) = (i \oplus 1, j \oplus 1)) &= 1/4\,. \end{aligned}$$

Let $D_t = Y_t \oplus (-X_t)$. Then $D_t$ behaves like a one-dimensional random walk on the set $\{0, 1, \ldots, n\}$, with absorbing barriers at $0$ and $n$. This walk moves left with probability $1/4$, right with probability $1/4$ and stays with probability $1/2$ in each step.

Looking back to our analysis of RANDOM 2-SAT, we can see that the expected time to reach a barrier is at most $2n^2$. The probability not to reach a barrier after $4n^2$ steps is at most $1/2$ by Markov's inequality. Thus after $4kn^2$ steps the probability not to reach a barrier is at most $2^{-k}$. We deduce that $\mathbb{P}(X_t \neq Y_t) \leq \varepsilon$ if $t \geq 4 \log_2(1/\varepsilon) n^2$. The required bound on the mixing time now follows from the Coupling Lemma.

**Example: Binary trees**

Consider the lazy random walk on the complete binary tree of $n = 2^{k+1} - 1$ vertices; with probability $1/2$, the particle stays at the same vertex and with probability $1/2$, it moves to a neighbour chosen u.a.r. We already know the unique stationary distribution from our analysis of random walks on graphs (clearly, this graph is bipartite, but the self-loops make the random walk aperiodic). We want to find an upper bound $\tau(\varepsilon)$ on the mixing time.

We will use the following coupling. As long as the two particles are not at the same level, at each step select exactly one of the two particles (u.a.r.) and move it to a neighbour u.a.r.

Observe that this is consistent with the transition probabilities of the lazy random walk. When they reach the same level, we change the coupling so that they will remain at the same level. With probability $1/2$ they both stay still. Otherwise, they both go up, or they both go down (making the same choice about which child to go to). Overall, we couple the two chains in two stages: first couple the levels and then the states themselves.

By the time that the chain starting closer to the root reaches a leaf, the levels of the two chains must have coupled. When it later reaches the root, the states also must have coupled. The expected time for this is at most the commute time between the root and the leaves, where the commute time between a vertex $u$ and a vertex $v$ is the expected time it takes, starting at $u$ to visit $v$ and then return to $u$.

If we go back to our analysis of the expected hitting time $h_{u,v}$ from a node $u$ to a neighbour $v$ we find that $h_{u,v} = O(n)$ so the commute time (from $u$ to $v$ and back to $u$) is twice this, which is also $O(n)$. Finally, the commute time between the root and some fixed leaf is $O(nk) = O(n \log n)$.

We have shown that there is some $T(n) = O(n \log n)$ such that the expected coupling is at most $T(n)$ (for any two places that the particles may have started). By Markov's inequality, the probability of not coupling in $2T(n)$ steps is at most $1/2$.

If we repeat this experiment $j = \log_2(1/\varepsilon)$ times, the probability of not coupling is at most $1/2^j \leq \varepsilon$. By the coupling lemma, the mixing time is at most $O(n \log(n) \log(1/\varepsilon))$.

**Sources**

- Mitzenmacher and Upfal, 2nd edition, Section 12.1-12.4
- Jerrum, Counting Sampling and Integrating: Algorithms and Complexity, Sections 4.1–4.2

## 4.6 Finding Couplings and Path Coupling

Suppose that we have a coupling $(X_t, Y_t)$ of a Markov chain $(M_t)$ with finite state space $\Omega$, transition matrix $P$, and a unique stationary distribution $\pi$.

An integral distance metric $d$ on $\Omega$ is a function $d\colon \Omega \times \Omega \to \mathbb{Z}_{\geq 0}$ such that, for all $x, y, z \in \Omega$,

- $d(x, y) = 0$ iff $x = y$ (the distance from a state to itself is zero, but all other distances are positive)
- $d(x, y) = d(y, x)$ (the distance function is symmetric)
- $d(x, z) \leq d(x, y) + d(y, z)$ (the distance function satisfies the triangle inequality)

By Markov's inequality,

$$\mathbb{P}(X_t \neq Y_t) = \mathbb{P}(d(X_t, Y_t) \geq 1) \leq \mathbb{E}[d(X_t, Y_t)].$$

Therefore, to bound the mixing time, we just need to show that $\mathbb{E}[d(X_t, Y_t)]$ goes quickly to $0$ as $t$ grows. The following lemma is useful.

**Lemma 4.6.1.** *(Coupling contraction lemma) Let $(X_t, Y_t)$ be a coupling of a Markov chain $(M_t)$ with a finite state space $\Omega$ and a unique stationary distribution. Let $d$ be an integral distance metric on $\Omega$ and let $D = \max_{(x,y)\in\Omega^2} d(x,y)$. If there is a $\beta \in (0,1)$ such that*

$$\mathbb{E}[d(X_1, Y_1) \mid (X_0, Y_0) = (x, y)] \leq \beta d(x, y) \tag{4.3}$$

*then the mixing time $\tau(\varepsilon)$ of $(M_t)$ satisfies $\tau(\varepsilon) \leq \lceil \ln(D/\varepsilon)/\ln(1/\beta) \rceil$.*

*Proof.* We will first show, by induction on $t$, that $\mathbb{E}[d(X_t, Y_t) \mid (X_0, Y_0) = (x, y)] \leq \beta^t d(x, y)$. The base case is $t = 0$. For $t > 0$,

$$\begin{aligned}\mathbb{E}[d(X_t, Y_t) \mid (X_0, Y_0) = (x, y)] &\leq \sum_{x',y'} \mathbb{P}((X_{t-1}, Y_{t-1}) = (x', y') \mid (X_0, Y_0) = (x, y))\, \beta d(x', y') \\ &= \beta\, \mathbb{E}[d(X_{t-1}, Y_{t-1}) \mid (X_0, Y_0) = (x, y)],\end{aligned}$$

so the claim follows by induction. By the definition of $D$, $\beta^t d(x, y) \leq \beta^t D$. So for $t \geq \ln(D/\varepsilon)/\ln(1/\beta)$ we have $\mathbb{E}[d(X_t, Y_t) \mid (X_0, Y_0) = (x, y)] \leq \varepsilon$. □

As an example, consider the following Markov chain. Consider the empty graph consisting of a set $V$ of $n$ vertices, and no edges between them. Let $q$ be some fixed number of colours. A state in $\Omega$ is an assignment $\sigma \colon V \to \{1, \ldots, q\}$ which assigns one of the $q$ colours to each vertex. The Markov chain is a sequence $X_0, X_1, \ldots$ of states in $\Omega$ where the transition from $X_t$ to $X_{t+1}$ is given as follows.

- *Choose $v$ u.a.r. from $V$*
- *Choose $c$ u.a.r. from $\{1, \ldots, q\}$*
- *With probability $1/2$, construct $X_{t+1}$ from $X_t$ by assigning vertex $v$ to colour $c$.*
- *Else $X_{t+1} = X_t$.*

The chain is obviously finite, irreducible and aperiodic, so it is ergodic and has a unique stationary distribution. Let $(X_t, Y_t)$ be the coupling that chooses the same vertex $v$ and colour $c$ in each copy and makes the same random choice about whether to colour $v$ with $c$ (rather than staying still). Let $d(\sigma, \tau)$ be the number of vertices on which the assignments $\sigma$ and $\tau$ disagree.

Now suppose $(X_0, Y_0) = (\sigma, \tau)$ with $d(\sigma, \tau) = k$. With probability $(k/n)$ a disagreeing vertex is chosen so with probability $(k/(2n))$ the number of disagreements goes down by 1. Thus, $\mathbb{E}[d(X_1, Y_1) \mid (X_0, Y_0) = (\sigma, \tau)] = d(\sigma, \tau) - d(\sigma, \tau)/(2n) = \beta d(\sigma, \tau)$ for $\beta = 1 - 1/(2n)$. Note that $D = n$. By the Lemma, $\tau(\varepsilon) = O(\ln(n/\varepsilon)/\ln(1/\beta))$. But $1/\beta = 1 + 1/(2n-1)$ and, for $x \in (0, 1)$, $\ln(1 + x) \geq x/2$, so $\ln(1/\beta) \geq 1/(2(2n-1))$ and $\tau(\varepsilon) = O(n \ln(n/\varepsilon))$.

### 4.6.1 Path Coupling

Finding couplings can be difficult, so we'll next see the *path coupling* technique, which makes it easier. The idea is that, instead of proving the contraction condition (4.3) for *every* pair of states $(x, y)$ we instead just prove it for a subset of $\Omega \times \Omega$, and this implies contraction for all pairs of states.

We use the following definitions. An *edge-weighted* graph for the state space $\Omega$ is a connected graph $H$ whose vertex set is $\Omega$ together with a positive integer distance $d(x,y)$ for each edge of $H$. We say that $(H,d)$ is *minimal* if, for every edge $(x,y)$, $d(x,y)$ is the length of a shortest path from $x$ to $y$ in $H$.

The reason that we want $(H,d)$ to minimal is that we want to use it to obtain an integral distance metric on $\Omega$. It is easy to turn an edge-weighted graph into a minimal one — if $d(x,y)$ is not the shortest-path distance from $x$ to $y$ in $H$, just remove $(x,y)$ from the edge set — this obviously will not disconnect $H$ since there is another (shorter) path from $x$ to $y$.

Given a minimal edge-weighted graph $(H,d)$ for $\Omega$, the corresponding integral distance metric on $\Omega$ is just the shortest-path metric on $H$. It won't cause any confusion to also call this $d$, so for all $(x,y) \in \Omega \times \Omega$, we define $d(x,y)$ to be the length of a shortest path from $x$ to $y$ in $H$.

To simplify the notation below, for every pair $(x,y)$ of distinct states in $\Omega$, we identify some particular shortest path $x = z_0, \ldots, z_\ell = y$ from $x$ to $y$ in $H$ and we call this $\text{path}(x,y)$.

The following lemma shows that if the we can define a coupling for which the contraction condition (4.3) holds for edges of $H$ then we can extend it to a coupling that satisfies the contraction condition for all pairs in $\Omega^2$.

**Lemma 4.6.2.** *Let $(M_t)$ be a Markov chain with finite state space $\Omega$. Let $(H,d)$ be a minimal edge-weighted graph for $\Omega$. Fix $\beta \in (0,1)$. Suppose that there is a coupling $(X_t,Y_t)$ of $(M_t)$ such that, for all edges $(x,y)$ of $H$, we have $\mathbb{E}[d(X_1,Y_1) \mid (X_0,Y_0) = (x,y)] \le \beta d(x,y)$. Then there is a coupling $(\widehat{X}_t,\widehat{Y}_t)$ of the chain such that, for all pairs $(x,y)$ in $\Omega^2$, we have $\mathbb{E}[d(\widehat{X}_1,\widehat{Y}_1) \mid (\widehat{X}_0,\widehat{Y}_0) = (x,y)] \le \beta d(x,y)$.*

*Proof.* Let $P$ be the transition matrix of the chain so that $P_{x,y}$ is the probability of moving from $x$ to $y$. Let $Q$ be the transition matrix of the coupling that contracts on edges so that for edges $(x,y)$, $Q_{(x,y),(x',y')}$ is the probability of moving from $(x,y)$ to $(x',y')$.

Now we are going to use $P$ and $Q$ to define a new coupling — we'll use $\widehat{Q}$ for the transition matrix of this new coupling, so that $\widehat{Q}_{(x,y),(x',y')}$ will be the probability of moving from $(x,y)$ to $(x',y')$.

If $x = y$ then from the definition of coupling, we have no choice. $\widehat{Q}_{(x,x),(x',x')} = P_{x,x'}$.

So suppose $x \neq y$ and consider $\text{path}(x,y)$, which is $x = z_0, \ldots, z_\ell = y$. We define the coupling from $(x,y)$ by first taking one step of the coupling $Q$ from $(z_0,z_1)$. Let the resulting random variables be $(Z_0',Z_1')$. Then for each $i \in 1,\ldots,\ell-1$, in order, we do the following. Conditioned on the value of $Z_i'$, we take one step of the coupling from $(z_i,z_{i+1})$ to obtain $(Z_i',Z_{i+1}')$. In the constructed coupling, $(x,y) = (z_0,z_\ell)$ goes to $(Z_0',Z_\ell')$.

If $\ell = 1$ then the constructed coupling just follows $Q$, so $\widehat{Q}_{(x,y),(x',y')} = Q_{(x,y),(x',y')}$.

If $\ell > 1$ then we can write down the relevant transition probability of the coupling as

$$\widehat{Q}_{(z_0,z_\ell),(z_0',z_\ell')} = \sum_{z_1',\ldots,z_{\ell-1}'} Q_{(z_0,z_1),(z_0',z_1')} \prod_{i=1}^{\ell-1} \frac{Q_{(z_i,z_{i+1}),(z_i',z_{i+1}')}}{P_{z_i,z_i'}}.$$

It is not hard to see that this does define a coupling — the main thing to check is that the transition from $z_\ell$ to $Z_\ell'$ is faithful. To see that it is, note that the probability that any particular $z_1'$ is chosen is $P_{z_1,z_1'}$ since $Q$ is faithful. For any $i \in \{1,\ldots,\ell-1\}$, choosing $Z_i'$ from $P_{z_i,z_i'}$ and then applying $Q$ to $(z_i,z_{i+1})$ conditioned on the value of $Z_i'$ is a faithful way to choose the value of $Z_{i+1}'$. (This can be made rigorous using the definition of $\widehat{Q}$.)

The main point for us is that the new coupling has the desired contraction. To see this, note that $\mathbb{E}[d(X_1, Y_1) \mid (X_0, Y_0) = (x, y)] = \mathbb{E}[d(Z'_0, Z'_\ell)]$. By the triangle inequality,

$$\mathbb{E}[d(Z'_0, Z'_\ell)] \leq \mathbb{E}\left[\sum_{i=0}^{\ell-1} d(Z'_i, Z'_{i+1})\right]$$

But by contraction on edges, this is at most $\beta \sum_{i=0}^{\ell-1} d(z_i, z_{i+1}) = \beta d(x, y)$.

□

Now, combining this lemma with the Coupling contraction lemma, we immediately get

**Lemma 4.6.3.** *(Path coupling) Let $(M_t)$ be a Markov chain with finite state space $\Omega$ and a unique stationary distribution. Let $(H, d)$ be a minimal edge-weighted graph for $\Omega$. Fix $\beta \in (0, 1)$. Suppose that there is a coupling $(X_t, Y_t)$ of $(M_t)$ such that, for all edges $(x, y)$ of $H$, we have $\mathbb{E}[d(X_1, Y_1) \mid (X_0, Y_0) = (x, y)] \leq \beta d(x, y)$. Let $D = \max_{(x,y)\in\Omega^2} d(x, y)$. Then the mixing time $\tau(\varepsilon)$ of $(M_t)$ satisfies $\tau(\varepsilon) \leq \lceil \ln(D/\varepsilon)/\ln(1/\beta) \rceil$.*

### 4.6.2 Example: Sampling Proper Colourings

A proper $q$-colouring of an $n$-vertex graph $G = (V, E)$ is an assignment $\sigma \colon V \to \{1, \ldots, q\}$ such that every edge $(u, v) \in E$ has $\sigma(u) \neq \sigma(v)$.

We will assume that the graph $G$ has maximum degree at most $\Delta$ and that $q \geq 2\Delta + 1$.

We will consider the following Markov Chain for sampling proper colourings. The Markov chain is a sequence $X_0, X_1, \ldots$ of states from the set of proper $q$-colourings of $G$. The transition from $X_t$ to $X_{t+1}$ is given as follows.

- *Choose $v$ u.a.r. from $V$*
- *Choose $c$ u.a.r. from $\{1, \ldots, q\}$*
- *If a neighbour of $v$ has colour $c$ in $X_t$ then $X_{t+1} \leftarrow X_t$.*
- *Otherwise, construct $X_{t+1}$ from $X_t$ by assigning vertex $v$ to colour $c$.*

The chain is aperiodic since there is a possibility of self-loop. It is irreducible since $q \geq \Delta + 2$. To see this, label the vertices as $v_1, \ldots, v_n$. Consider some proper colouring $\sigma$. To move from any proper colouring to $\sigma$, update the vertices one-by-one in the order $v_1, \ldots, v_n$. To update vertex $v_i$, first move any neighbours of $v_i$ in $v_{i+1}, \ldots, v_n$ to colours other than the (at most $\Delta - 1$) colours of their other neighbours, or the current or new colour of $v_i$. Then recolour $v_i$. Thus, the chain is ergodic, and has a unique stationary distribution $\pi$. By detailed balance, $\pi$ is uniform on the proper $q$-colourings of $G$.

We'd like to use path coupling to show that the chain is rapidly mixing. It is going to be convenient to have the distance metric $d$ as simple as possible, and for this it will be useful to make the state space $\Omega$ be the set of *all* colourings of $G$ (including improper colourings). It is easy to see that the chain still has a unique stationary distribution $\pi$, even with this enlarged state space. The unique stationary distribution $\pi$ is the uniform distribution on proper colourings. (Each time a vertex is chosen, it has a chance to obtain a colour that differs from the colours of its neighbours. Once this happens for all vertices, the chain reaches

a proper colouring. Once it visits a proper colouring, the chain stays within the set of proper colourings.)

Now let $H$ be the graph on $\Omega$ in which there is an edge from $\sigma$ to $\tau$ if and only if $\sigma$ and $\tau$ differ on the colour of exactly one vertex. For every edge $(\sigma, \tau)$, let $d(\sigma, \tau) = 1$. Since $\Omega$ contains all colourings, the corresponding distance metric $d$ is just Hamming distance. So for every pair of states $(\sigma, \tau)$, $d(\sigma, \tau)$ is the number of vertices on which $\sigma$ and $\tau$ assign different colours.

The next step is to define a coupling $(X_t, Y_t)$ such that, for some $\beta \in (0, 1)$ and all edges $(\sigma, \tau)$, we have $\mathbb{E}[d(X_1, Y_1) \mid (X_0, Y_0) = (\sigma, \tau)] \leq \beta d(x, y)$. The coupling will be as follows, starting from edge $(X_t, Y_t)$.

- *Choose $v$ u.a.r. from $V$.*
- *Choose $c$ u.a.r. from $\{1, \ldots, q\}$.*
- *If $v$ has a neighbour $w$ with $X_t(w) \neq Y_t(w)$ (* there is at most one such $w$ since $(X_t, Y_t)$ is an edge *)*
  - *If $c = X_t(w)$ then use $(v, X_t(w))$ to update $X_{t+1}$ and use $(v, Y_t(w))$ to update $Y_{t+1}$ (following the Markov chain in both cases)*
  - *If $c = Y_t(w)$ then use $(v, Y_t(w))$ to update $X_{t+1}$ and use $(v, X_t(w))$ to update $Y_{t+1}$*
  - *Else use $v$ and $c$ to update $X_{t+1}$ and $Y_{t+1}$*
- *Else use $v$ and $c$ to update $X_{t+1}$ and $Y_{t+1}$*

It is easy to see that the coupling is faithful (the projection onto the $X$ copy mimics the behaviour of the original chain, and the projection onto the $Y$ copy merely (in some cases) swaps the random choice of two colours).

So take $t = 0$ and suppose that $(X_0, Y_0) = (\sigma, \tau)$ where $(\sigma, \tau)$ is an edge of $H$ so $\sigma$ and $\tau$ differ on a single vertex (let's call it $z$). Let $\Gamma(z)$ be the neighbours of $z$ in $G$. There are only two choices of $(v, c)$ that cause $d(X_1, Y_1)$ to differ from $d(X_0, Y_0)$, so we consider these.

- If $v = z$ and $c \notin X_0(\Gamma(z))$ then the colour of $z$ is replaced with $c$ in both copies, so $X_1 = Y_1$ and $d(X_1, Y_1) = 0$. Since there are at least $q - \Delta$ colours that are not in $X_0(\Gamma(z))$, the probability that this happens is at least $\frac{1}{n}\frac{q-\Delta}{q}$.
- If $v = u$ for some $u \in \Gamma(z)$ and $c = Y_0(z)$ then it is possible that $X_1(u) \neq Y_1(u)$. But in any case we have $d(X_1, Y_1) \leq 2$ since $d$ is Hamming distance. Since $|\Gamma(z)| \leq \Delta$, the probability that this happens is at most $\Delta/(nq)$.

Adding these up, and noting that $d(X_0, Y_0) = 1$,

$$\mathbb{E}[d(X_1, Y_1)] \leq 1 - \frac{q-\Delta}{nq} + \frac{\Delta}{nq} = d(X_0, Y_0)\left(1 - \frac{q - 2\Delta}{nq}\right) \leq d(X_0, Y_0)\left(1 - \frac{1}{nq}\right)$$

Now we can apply the path coupling lemma with $\beta = \left(1 - \frac{1}{nq}\right)$ and $D = n$ (since the maximum Hamming distance is $n$). This shows that the mixing time satisfies $\tau(\varepsilon) = O(\ln(n/\varepsilon)/\ln(1/\beta))$. Using the same argument that we used for the edge-free case, $\tau(\varepsilon) = O(n \ln(n/\varepsilon))$.

**Sources**

- Mitzenmacher and Upfal, 2nd edition, Section 12.5-12.6
- Jerrum, Counting Sampling and Integrating: Algorithms and Complexity, Section 4.3
- Dyer and Greenhill, Random Walks on Combinatorial Objects

PART 5
MARTINGALES

## 5.1 Martingales and stopping times

### 5.1.1 Conditional Expectation

First, a quick recap.

- A *random variable* $X$ on a sample space $\Omega$ is a real-valued measurable function $X : \Omega \to \mathbb{R}$.
- Conditional probability: $\mathbb{P}(X = x \mid Y = y) = \mathbb{P}(X = x \wedge Y = y)/\mathbb{P}(Y = y)$
- Conditional expectation: $\mathbb{E}[X \mid Y = y] = \sum_x x\mathbb{P}(X = x \mid Y = y)$
- Easy facts
  - $\mathbb{E}[X] = \sum_y \mathbb{P}(Y = y)\,\mathbb{E}[X \mid Y = y]$
  - $\mathbb{E}[X + Z \mid Y = y] = \mathbb{E}[X \mid Y = y] + \mathbb{E}[Z \mid Y = y]$

What is $\mathbb{E}[X \mid Y]$? It is a random variable! It is a function of $Y$. Whenever $Y = y$, it takes that value $\mathbb{E}[X \mid Y = y]$.

The following fact is easy, but important. $\mathbb{E}[X] = \mathbb{E}[\mathbb{E}[X \mid Y]]$.

To see this, write out the right-hand-side. It is $\sum_y \mathbb{P}(Y = y)\,\mathbb{E}[X \mid Y = y]$. Then use the "easy fact" above.

### 5.1.2 Martingales

**Definition 5.1.1.** A martingale is a sequence of random variables $X_0, X_1, \ldots$, such that each $X_n$ has bounded expectation (meaning that $\mathbb{E}[|X_n|] < \infty$) and, for every $n \geq 0$,

$$\mathbb{E}[X_{n+1} \,|\, X_0, \ldots, X_n] = X_n.$$

This is generalised as follows.

**Definition 5.1.2.** A sequence of random variables $Z_0, Z_1, \ldots$ is a martingale with respect to the sequence $X_0, X_1, \ldots$ if for all $n \geq 0$ the following conditions hold:

- $Z_n$ is a function of $X_0, \ldots, X_n$,
- $\mathbb{E}[|Z_n|] < \infty$,

- $\mathbb{E}[Z_{n+1} \mid X_0, \ldots, X_n] = Z_n$.

If we replace the last condition by $\mathbb{E}[Z_{n+1} \mid X_0, \ldots, X_n] \leq Z_n$, we get a *supermartingale*. If we replace the last condition by $\mathbb{E}[Z_{n+1} \mid X_0, \ldots, X_n] \geq Z_n$, we get a *submartingale*.

### Gambler's fortune

A gambler plays a sequence of *fair* games. Let $X_i$ be the amount that the gambler wins on the $i$'th game (this can be positive or negative). What we mean by "fair" is that $\mathbb{E}[X_i] = 0$. Let $Z_i$ be the gambler's total winnings at the end of the $i$'th game, so $Z_i = X_0 + \cdots + X_i$.

We'll assume that the fair games have the property that $\mathbb{E}[|X_i|] < \infty$. From this it is easy to check that $\mathbb{E}[|Z_i|] = \mathbb{E}[|X_0 + \cdots + X_i|] \leq \sum_{j=0}^{i} \mathbb{E}[|X_j|] < \infty$. To see that $Z_0, Z_1, \ldots$ is a martingale with respect to $X_0, X_1, \ldots$ we note that

$$\mathbb{E}[Z_{i+1} \mid X_0, \ldots, X_i] = \mathbb{E}[X_{i+1} + Z_i \mid X_0, \ldots, X_i] = Z_i.$$

Note that our only assumptions were that $\mathbb{E}[X_i] = 0$ and $\mathbb{E}[|X_i|] < \infty$. We still have a martingale if $X_i$ depends on $X_0, \ldots, X_{i-1}$ (so we still have a martingale even if the gambler decides how much to bet based on the outcomes of previous games).

### Balls and bins

Suppose that we throw $m$ balls into $n$ bins independently and u.a.r. This is one of the most-studied random experiments and we usually ask questions about the expected maximum load and about the expected number of empty bins.

Here we consider the expected number of empty bins. Let $X_i$ be the random variable representing the bin into which the $i$'th ball falls. Let $Y$ be a random variable representing the number of empty bins at the end. We will show that the sequence of random variables

$$Z_i = \mathbb{E}[Y \mid X_1, \ldots, X_i]$$

is a martingale. The first step is to check that $Z_i$ is a function of $X_1, \ldots, X_i$, which is clearly true.

It is not very important that the sequences $X_1, \ldots, X_m$ and $Z_1, \ldots, Z_m$ have indices that start from 1 rather than 0. To fit the formal definition, we can take $Z_0$ and $X_0$ to be constants. In particular, we can take $X_0 = 0$ and $Z_0 = \mathbb{E}[Y] = \mathbb{E}[Z_1]$. (In the future, we will index martingale sequences by 0, or 1, or something else — whatever is most convenient.)

To see that we have a martingale, note that $\mathbb{E}[|Z_i|] \leq n$. Also,

$$\begin{aligned}
\mathbb{E}[Z_{i+1} \mid X_1, \ldots, X_i] &= \mathbb{E}[\mathbb{E}[Y \mid X_1, \ldots, X_i, X_{i+1}] \mid X_1, \ldots, X_i] \\
&= \mathbb{E}[Y \mid X_1, \ldots, X_i] \\
&= Z_i.
\end{aligned}$$

We can view $Z_i$ as an estimate of $Y$ after having observed the outcomes $X_1, \ldots, X_i$. At the beginning $Z_0$ is a crude estimate, simply the expectation of $Y$. As we add more balls to the bins, $Z_i$'s give improved estimates of $Y$, and at the end we get the exact value $Z_m = Y$.

### 5.1.3 Doob martingales

The balls and bins example is a typical *Doob martingale*. In general, Doob martingales are processes in which we obtain a sequence of improved estimates of the value of a random variable as information about it is revealed progressively.

In general, to define a Doob martingale, we have a sequence $X_0, \ldots, X_m$ of random variables and a random variable $Y$, depending on $X_0, \ldots, X_m$. We define

$$Z_i = \mathbb{E}[Y \mid X_0, \ldots, X_i].$$

$Y$ is chosen so that $\mathbb{E}[|Z_i|] < \infty$. Then we conclude that $Z_0, Z_1, \ldots$ is a martingale with respect to $X_0, X_1, \ldots$ since

$$\begin{aligned}\mathbb{E}[Z_{t+1} \mid X_0, \ldots, X_t] &= \mathbb{E}[\mathbb{E}[Y \mid X_0, \ldots, X_t, X_{t+1}] \mid X_0, \ldots, X_t]] \\ &= \mathbb{E}[Y \mid X_0, \ldots, X_t] \\ &= Z_t.\end{aligned}$$

### 5.1.4 Stopping times

**Definition 5.1.3.** A nonnegative, integer-valued random variable $T$ is a *stopping time* for the sequence $Z_0, Z_1, \ldots$ if, for every $n$, the event $T = n$ depends only on $Z_0, Z_1, \ldots, Z_n$.

In the gambler's fortune example, one possible stopping time would be the smallest $i$ such that $Z_i \geq k$ (for some fixed value $k$).

If instead, the gambler fixes some maximum number of games $n$, and then wants to stop whenever he receives the maximum winnings during the first $n$ games, this is not a stopping time.

If he decides to stop after exactly $t$ games, this is a stopping time.

#### Stopping times and martingales

**Lemma 5.1.4.** *Let $Z_0, \ldots, Z_t$ be a martingale with respect to $X_0, \ldots, X_t$. Then*

$$\mathbb{E}[Z_t] = \mathbb{E}[Z_0].$$

*Proof.* From the definition of martingales, $\mathbb{E}[Z_{i+1} \mid X_0, \ldots, X_i] = Z_i$. The lemma follows by taking expectations on both sides to get $\mathbb{E}[Z_{i+1}] = \mathbb{E}[Z_i]$ and applying induction. □

The question is whether the lemma applies not only at fixed times $t$ but at other stopping times. The answer is negative as the following example shows: Consider again the gambler's fortune and the first time that the gambler wins an amount $k > 0$. This is a stopping time but clearly the expectation at the end is $k$ which is greater than the expectation at the beginning which is 0. However under certain conditions the lemma can be generalised to stopping times.

**Theorem 5.1.5** (optional stopping theorem)**.** *Let $Z_0, Z_1, \ldots$ be a martingale with respect to $X_1, X_2, \ldots$. If $T$ is a stopping time for $X_1, X_2, \ldots$ and at least one of the following conditions holds*

- *$T$ is bounded,*

- *there is a constant $c$ such that, for every $i$, $|Z_{\min\{i,T\}}| \le c$, or*
- *$\mathbb{E}[T] < \infty$ and there is a constant $c$ such that, for every $i$,*

$$\mathbb{E}[|Z_{\min\{i+1,T\}} - Z_{\min\{i,T\}}| \,|\, X_1, \ldots, X_i] \le c,$$

*then $\mathbb{E}[Z_T] = \mathbb{E}[Z_0]$.*

*A similar conclusion holds for supermartingales, i.e., $\mathbb{E}[Z_T] \le \mathbb{E}[Z_0]$, and for submartingales, i.e., $\mathbb{E}[Z_T] \ge \mathbb{E}[Z_0]$.*

Note that the gambler's stopping rule of stopping when the gambler is ahead by $k$, i.e., when $Z_t \ge k$, does not satisfy any of the conditions. $T$ can be unbounded. Furthermore, $Z_{\min\{i,T\}}$ can be arbitrarily negative. We will see below (just before Section 5.1.5) that $\mathbb{E}[T]$ is also unbounded.

**Example:** Consider the random walk on the line. For $a, b > 0$, what is the probability that the random walk will reach $a$ before it reaches $-b$. This is equivalent to asking for the probability that a gambler wins if the rule is to quit the first time that the gambler is either ahead by an amount $a$ or behind by an amount $b$.

Let $X_i$, $i = 1, 2, \ldots$ be the amount won in the $i$'th game, $X_i \in \{-1, 1\}$. Then $Z_i = \sum_{j=1}^{i} X_j$ is a martingale with respect to $X_i$. The first time $T$ that satisfies $Z_T \in \{-b, a\}$ is a stopping time for $X_1, X_2, \ldots$. We can apply the optional stopping theorem because the second condition is satisfied — for every $i$, $|Z_{\min\{i,T\}}| \le \max\{a, b\}$. Thus, $\mathbb{E}[Z_T] = \mathbb{E}[Z_0] = 0$.

We can use this result to calculate the probability that the gambler wins. If $q$ is the probability of reaching $a$ before reaching $-b$, then $E[Z_T] = qa + (1 - q)(-b)$. Setting this to 0, we get

$$q = \frac{b}{a + b}.$$

**Example:** Consider again the random walk on the line of the previous example. We want to estimate the expected number of steps of the random walk to reach $a$ or $-b$. We define the random variable $Y_t = Z_t^2 - t$ and observe that it is a martingale with respect to $X_1, X_2, \ldots$:

$$\begin{aligned}
\mathbb{E}[Y_{t+1} \,|\, X_1, \ldots, X_t] &= \mathbb{E}[Z_{t+1}^2 \,|\, X_1, \ldots, X_t] - (t + 1) \\
&= \frac{1}{2}(Z_t + 1)^2 + \frac{1}{2}(Z_t - 1)^2 - (t + 1) \\
&= Z_t^2 - t \\
&= Y_t.
\end{aligned}$$

We can apply the optional stopping theorem because the third condition holds. First, it is fairly easy to see that $\mathbb{E}[T] < \infty$ (why?). Also,

$$\mathbb{E}[|Y_{\min\{i+1,T\}} - Y_{\min\{i,T\}}| \,|\, X_1, \ldots, X_i] \le 2 \max\{a, b\}.$$

Therefore $\mathbb{E}[Y_T] = \mathbb{E}[Y_0] = 0$.

But $\mathbb{E}[Y_T] = \mathbb{E}[Z_T^2] - \mathbb{E}[T]$. We can compute $\mathbb{E}[Z_T^2]$ using the probabilities from the previous example:

$$\mathbb{E}[Z_T^2] = \frac{b}{a + b} a^2 + \frac{a}{a + b} b^2 = ab,$$

to find $\mathbb{E}[T] = ab$.

Note that by taking an arbitrarily high value for $b$, this says that the expected time to reach $a$ is unbounded. This explains why we cannot apply the optional stopping theorem for the gambler's stopping rule "quit when you are ahead an amount $a$".

### 5.1.5 Wald's Equation

**Theorem 5.1.6** (Wald's equation)**.** *Let $X_1, X_2, \ldots$ be nonnegative, independent, identically distributed random variables with distribution $X$. Let $T$ be a stopping time for this sequence. If $T$ and $X$ have bounded expectation, then*

$$\mathbb{E}\left[\sum_{i=1}^{T} X_i\right] = \mathbb{E}[T] \cdot \mathbb{E}[X].$$

*Proof.* We will apply the optional stopping theorem to the sequence

$$Z_t = \sum_{i=1}^{t} (X_i - \mathbb{E}[X]).$$

First, we check that $Z_1, Z_2, \ldots$ is a martingale with respect to $X_1, X_2, \ldots$. We have assumed that $X$ has bounded expectation, so $Z_i$ does. Then $\mathbb{E}[Z_{t+1} \mid X_1, \ldots, X_t] = \mathbb{E}[X_{t+1} - \mathbb{E}[X] + Z_t \mid X_1, \ldots, X_t] = Z_t$.

Now to apply the optimal stopping theorem we will check that condition 3 holds. It suffices to show that $\mathbb{E}[|Z_{i+1} - Z_i| \mid X_1, \ldots, X_i]$ is bounded. This is $\mathbb{E}[|X_{i+1} - \mathbb{E}[X]|]$, which is at most $2\,\mathbb{E}[X]$. Thus, by optional stopping, $\mathbb{E}[Z_T] = \mathbb{E}[Z_1] = 0$.

So it suffices to show that $\mathbb{E}[Z_T] = \mathbb{E}\left[\sum_{i=1}^{T} X_i\right] - \mathbb{E}[T] \cdot \mathbb{E}[X]$, which follows by linearity of expectation.

□

**Example:** We roll a fair die until we get the first 5. What is the expected sum of all rolls? Let $X_t \in \{1, \ldots, 6\}$ be the outcome of the $t$'th roll, let $T$ be the time when the first 5 appears and let's apply Wald's equation. The expected sum is $\mathbb{E}[T] \cdot \mathbb{E}[X] = 6 \cdot 7/2 = 21$, because the expected number of steps $\mathbb{E}[T]$ until we see the first 5 is 6, and $\mathbb{E}[X_i] = 7/2$. Notice that we get the same total when we stop at the first 1 (or any other value for that matter).

#### Sources

- Mitzenmacher and Upfal, 2nd edition, Sections 2.3, 13.1–13.2

A note about sources: Theorem 5.1.5 is slightly different to the one in the book — we have replaced each instance of "$i$" with "$\min\{i, T\}$". This is without loss of generality. Suppose that $Z_0, Z_1, \ldots$ is a martingale with respect to $X_1, X_2, \ldots$ and that $T$ is a stopping time for $X_1, X_2, \ldots$. Let $\hat{Z}_0, \hat{Z}_1, \ldots$ be the sequence given by $\hat{Z}_i = Z_{\min\{i,T\}}$. It is easy to see that $\hat{Z}_0, \hat{Z}_1, \ldots$ is a martingale (since $Z_0, Z_1, \ldots$ is). If the martingale $\hat{Z}_0, \hat{Z}_1, \ldots$ satisfies one of the conditions in the optional stopping theorem in the book, then we can conclude that $\mathbb{E}[Z_T] = \mathbb{E}[\hat{Z}_T] = \mathbb{E}[\hat{Z}_0] = \mathbb{E}[Z_0]$. Thus, for convenience, we have incorporated the use of this trick into the statement of the theorem.

## 5.2 The Azuma-Hoeffding inequality

Chernoff bounds are almost tight for most purposes for sums of independent 0-1 random variables, but they cannot be used for sums of dependent variables. In the dependent case, if the random variables come from a martingale, the Azuma-Hoeffding inequality provides a more general, albeit weaker, concentration bound.

**Theorem 5.2.1** (Azuma-Hoeffding inequality)**.** *Let $X_0, X_1, \ldots, X_n$ be a martingale such that*

$$|X_i - X_{i-1}| \leq c_i.$$

*Then for any $\lambda > 0$,*

$$\mathbb{P}(X_n - X_0 \geq \lambda) \leq \exp\left(-\frac{\lambda^2}{2\sum_{i=1}^n c_i^2}\right), \text{ and}$$
$$\mathbb{P}(X_n - X_0 \leq -\lambda) \leq \exp\left(-\frac{\lambda^2}{2\sum_{i=1}^n c_i^2}\right).$$

*Proof.* We will only prove the first inequality, as the proof of the second one is very similar. The proof is again an application of Markov's inequality to an appropriate random variable and it is similar to the proof of Chernoff's bounds.

To simplify the notation, we use the variables $Y_i = X_i - X_{i-1}$. The steps of the proof are

1. - We use the standard technique of Chernoff bounds and instead of bounding $\mathbb{P}(X_n - X_0 \geq \lambda)$, we bound $\mathbb{P}(e^{t(X_n - X_0)} \geq e^{\lambda t})$ using Markov's inequality

$$\mathbb{P}(e^{t(X_n - X_0)} \geq e^{\lambda t}) \leq e^{-\lambda t}\,\mathbb{E}[e^{t(X_n - X_0)}].$$

   - From now on we focus on $\mathbb{E}[e^{t(X_n - X_0)}]$, which can be rewritten in terms of $Y_i$'s instead of $X_i$'s, as

$$\mathbb{E}[e^{t(X_n - X_0)}] = \mathbb{E}\left[\prod_{i=1}^n e^{tY_i}\right],$$

   by telescoping, $X_n - X_0 = \sum_{i=1}^n (X_i - X_{i-1}) = \sum_{i=1}^n Y_i$.

2. At this point in the proof of the Chernoff bounds, we used the fact that the variables are independent and we rewrote the expectation of the product as a product of expectations. We cannot do this here because random variables $Y_i$ are not in general independent. Instead, we consider the conditional expectation

$$\mathbb{E}\left[\prod_{i=1}^n e^{tY_i} \mid X_0, \ldots, X_{n-1}\right] = \left(\prod_{i=1}^{n-1} e^{tY_i}\right) \mathbb{E}\left[e^{tY_n} \mid X_0, \ldots, X_{n-1}\right],$$

   because for fixed $X_0, \ldots, X_{n-1}$, all but the last factor in the product are constants and can be moved out of the expectation.

   With this in mind, we turn our attention on finding an upper bound on $\mathbb{E}[e^{tY_i} \mid X_0, \ldots, X_{i-1}]$.

- We first observe that $\mathbb{E}[Y_i \mid X_0, \ldots, X_{i-1}] = 0$, by the martingale property:

$$\begin{aligned}\mathbb{E}[Y_i \mid X_0, \ldots, X_{i-1}] &= \mathbb{E}[X_i - X_{i-1} \mid X_0, \ldots, X_{i-1}] \\ &= \mathbb{E}[X_i \mid X_0, \ldots, X_{i-1}] - \mathbb{E}[X_{i-1} \mid X_0, \ldots, X_{i-1}] \\ &= X_{i-1} - X_{i-1} \\ &= 0\end{aligned}$$

- Using the premise that $|Y_i| \leq c_i$, we bound

$$e^{tY_i} \leq \beta_i + \gamma_i Y_i,$$

for $\beta_i = (e^{tc_i} + e^{-tc_i})/2 \leq e^{(tc_i)^2/2}$, and $\gamma_i = (e^{tc_i} - e^{-tc_i})/(2c_i)$. To show this, rewrite $Y_i$ as $Y_i = rc_i + (1-r)(-c_i)$, where $r = \frac{1+Y_i/c_i}{2} \in [0,1]$, and use the convexity of $e^{tx}$ to get

$$\begin{aligned}e^{tY_i} &\leq re^{tc_i} + (1-r)e^{-tc_i} \\ &= \frac{e^{tc_i} + e^{-tc_i}}{2} + Y_i \frac{e^{tc_i} - e^{-tc_i}}{2c_i} \\ &= \beta_i + \gamma_i Y_i.\end{aligned}$$

To bound $\beta_i$ from above, use the fact that for every $x$: $e^x + e^{-x} \leq 2e^{x^2/2}$.

- Combine the above to get

$$\begin{aligned}\mathbb{E}[e^{tY_i} \mid X_0, \ldots, X_{i-1}] &\leq \mathbb{E}[\beta_i + \gamma_i Y_i \mid X_0, \ldots, X_{i-1}] \\ &= \beta_i \leq e^{(tc_i)^2/2}.\end{aligned}$$

It follows that

$$\begin{aligned}\mathbb{E}\left[\left(\prod_{i=1}^{n} e^{tY_i}\right) \mid X_0, \ldots, X_{n-1}\right] &= \left(\prod_{i=1}^{n-1} e^{tY_i}\right) \mathbb{E}[e^{tY_n} \mid X_0, \ldots, X_{n-1}] \\ &\leq \left(\prod_{i=1}^{n-1} e^{tY_i}\right) e^{(tc_n)^2/2}.\end{aligned}$$

3. We now take expectations on both sides to get rid of the conditional expectation,

$$\mathbb{E}\left[\prod_{i=1}^{n} e^{tY_i}\right] \leq \mathbb{E}\left[\prod_{i=1}^{n-1} e^{tY_i}\right] e^{(tc_n)^2/2}.$$

4. Using standard techniques we can now finish the proof.

- By induction, $\mathbb{E}[\prod_{i=1}^{n} e^{tY_i}] \leq \prod_{i=1}^{n} e^{(tc_i)^2/2} = e^{t^2 \sum_{i=1}^{n} c_i^2/2}$
- Therefore $\mathbb{P}(e^{t(X_n - X_0)} \geq e^{\lambda t}) \leq e^{-\lambda t} e^{t^2 \sum_{i=1}^{n} c_i^2/2}$
- Set $t = \lambda / \sum_{i=1}^{n} c_i^2$ to minimise the above expression and get the bound of the theorem.

Step 2 in the proof is crucial because, using conditionals, it bounds the product of the random variables $\prod_{i=1}^{n-1} e^{tY_i}$ and $e^{tY_n}$, although the two variables are not in general independent. □

### 5.2.1 Gambler's fortune, concentration of gains

Consider again the case of a gambler who plays a sequence of *fair* games. We have seen that if $Z_i$ denotes the gambler's total winnings immediately after the $i$'th game, the sequence $Z_0, Z_1, \ldots, Z_n$ is a martingale. Suppose that the gambler has a very sophisticated algorithm to decide the amount of the bet every day that takes into account past bets and outcomes. Since the games are fair, the expected gain $Z_n$ is 0.

Are the winnings concentrated around the mean value? Not in general; consider for example, the case in which the gambler puts higher and higher bets. Suppose now that there is a bound on the size of bets, for example suppose that the bets are at most 10 pounds. By the Azuma-Hoeffding inequality, the final winnings are concentrated with high probability in $[-k, k]$, where $k = O(\sqrt{n \log n})$.

**Sources**

- Mitzenmacher and Upfal, 2nd edition, Sections 13.4
- James Lee, lecture7.pdf (page 1)

## 5.3 Applications of the Azuma-Hoeffding inequality

In many cases, it is easier to apply the Azuma-Hoeffding inequality in the following form, which is known as McDiarmid's inequality. Its proof is essentially an application of the Azuma-Hoeffding inequality using the Doob martingale $Z_t = \mathbb{E}[f(X_1, \ldots, X_n) \,|\, X_1, \ldots, X_t]$. Actually the constant in the exponent in the following theorem is slightly better than the one we can get by a direct application of the given Azuma-Hoeffding inequality.

**Theorem 5.3.1** (McDiarmid's inequality)**.** *Let $f$ be a $c$-Lipschitz function, that is, it satisfies the following Lipschitz condition for every $x_1, \ldots, x_n$, and for every $i$ and $x_i'$:*

$$|f(x_1, \ldots, x_{i-1}, x_i, x_{i+1}, \ldots, x_n) - f(x_1, \ldots, x_{i-1}, x_i', x_{i+1}, \ldots, x_n)| \leq c.$$

*Then for independent random variables $X_1, \ldots, X_n$ and any $\lambda > 0$:*

$$\mathbb{P}(|f(X_1, \ldots, X_n) - \mathbb{E}[f(X_1, \ldots, X_n)]| \geq \lambda) \leq 2e^{-2\lambda^2/(nc^2)}.$$

The following examples could be done by using McDiarmid's inequality, but we will write out the martingales.

### 5.3.1 Chromatic number of random graphs

Random graphs are very useful models. The standard model is the $G_{n,p}$ model in which we create a graph of $n$ vertices and edges selected independently, each with probability $p$. There are other models, such as the rich-get-richer models that try to model social networks, random geometric graphs, and others.

Let's consider the chromatic number $\chi(G)$ of a random graph $G$ in the $G_{n,p}$ model. The chromatic number of a graph is the minimum number of colours needed in order to colour the vertices of the graph in such a way that no adjacent vertices have the same colour. It is NP-hard to compute the chromatic number of a graph.

Interesting questions:

- What is the expected chromatic number $\mathbb{E}[\chi(G)]$?
- Is the chromatic number concentrated around its expected value?

Remarkably, we can answer the second question without knowing $\mathbb{E}[\chi(G)]$. To do this we consider a *vertex exposure martingale*. Let $G_i$ denote the subgraph with vertices $\{1, \ldots, i\}$ and consider the Doob martingale

$$Z_i = \mathbb{E}[\chi(G) \mid G_1, \ldots, G_i].$$

Note that $Z_0 = \mathbb{E}[\chi(G)]$.

By exposing a new vertex we cannot change the expected chromatic number by more than 1 so it is not too hard to show

$$|Z_{i+1} - Z_i| \leq 1.$$

We can then apply the Azuma-Hoeffding inequality to get

$$\mathbb{P}(|Z_n - Z_0| \geq \lambda\sqrt{n}) \leq 2e^{-\lambda^2/2}.$$

This shows that the chromatic number is with high probability within $O(\sqrt{n \log n})$ from its expected value[1].

### 5.3.2 Pattern matching

Consider a random string $X = (X_1, \ldots, X_n)$ in which the characters are selected independently and u.a.r. from a fixed alphabet $\Sigma$, with $|\Sigma| = s$. Let $p = (p_1, \ldots, p_k)$ be a fixed string (pattern). Let $F$ be the number of occurrences of $p$ in $X$.

- What is the expected number $\mathbb{E}[F]$? The probability that the pattern appears in positions $i+1, \ldots, i+k$ is exactly $1/s^k$. By linearity of expectations, the expected number of occurrences of $p$ in $X$ is $(n-k+1)/s^k$.

We will show that the number of occurrences of $p$ in $X$ is highly concentrated. Consider the Doob martingale

$$Z_i = \mathbb{E}[F \mid X_1, \ldots, X_i],$$

with $Z_0 = \mathbb{E}[F]$ and $Z_n = F$. The important observation is that

$$|Z_{i+1} - Z_i| \leq k,$$

which can be shown because every character can participate in at most $k$ occurrences of the pattern.

We can apply the Azuma-Hoeffding inequality to getting

$$\mathbb{P}(|F - \mathbb{E}[F]| \geq \lambda k\sqrt{n}) \leq 2\exp(-\lambda^2 k^2 n/(2nk^2) = 2e^{-\lambda^2/2}.$$

[1] In fact, much sharper concentration bounds are known.

### 5.3.3 Balls and bins - number of empty bins

We consider again the balls and bins experiment. Suppose that we throw $m = n$ balls into $n$ bins and let's consider the expected number of empty bins. Let $X_i$ be the random variable representing the bin into which the $i$'th ball falls. Let $Y$ be a random variable representing the number of empty bins and consider the Doob martingale

$$Z_i = \mathbb{E}[Y \mid X_1, \ldots, X_i],$$

with $Z_0 = \mathbb{E}[Y]$ and $Z_n = Y$.

Since each ball cannot change the expectation by more than 1 it is straightforward to show

$$|Z_{i+1} - Z_i| \leq 1.$$

Applying the Azuma-Hoeffding inequality, we again see that the number of empty bins is highly concentrated around its mean value. More precisely,

$$\mathbb{P}(|Y - \mathbb{E}[Y]| \geq \varepsilon n) \leq 2e^{-\varepsilon^2 n/2}.$$

But what is the expected number $\mathbb{E}[Y]$? The probability that a bin is empty is $(1-1/n)^n$, so by linearity of expectations, the expected number of empty bins is

$$n\left(1 - \frac{1}{n}\right)^n \approx \frac{n}{e}.$$

**Sources**

- Mitzenmacher and Upfal, 2nd edition, Sections 13.5
- James Lee, lecture7.pdf (page 1)

PART 6
ADDITIONAL TOPICS

## 6.1 The Lovász Local Lemma

Recall from Part 2 that the *probabilistic method* is a method used to show existence of an object with a certain property by showing that, with respect to some distribution over the objects, either

- a random object has the property with non-zero probability, or
- the expected number of objects with the property is positive.

A central tool for doing this is the Lovász Local Lemma. The setting of the lemma is as follows: there is a set of "bad events" $\{\mathcal{E}_1, \ldots, \mathcal{E}_n\}$ and we want to show that there is a state in the sample space that does not belong to any bad event. Equivalently, we want to show $\mathbb{P}(\bigwedge_{i=1}^n \overline{\mathcal{E}}_i) > 0$. If each bad event has probability strictly less than 1 and the events are mutually independent, then there is a good state in the sample space, because $\mathbb{P}(\bigwedge_{i=1}^n \overline{\mathcal{E}}_i) = \prod_{i=1}^n \mathbb{P}(\overline{\mathcal{E}}_i) > 0$. The Lovász Local Lemma extends this to the case in which the events have some limited dependency and the probability of each bad event is bounded away from 1. To state the lemma, let's first define the dependency graph of a set of events.

**Definition 6.1.1.** A dependency graph for a set of events $\{\mathcal{E}_1, \ldots, \mathcal{E}_n\}$ is a graph $G = (V, E)$ such that $V = \{\mathcal{E}_1, \ldots, \mathcal{E}_n\}$ and, for every $i \in \{1, \ldots, n\}$, event $\mathcal{E}_i$ is mutually independent of the events $\{\mathcal{E}_j \mid (\mathcal{E}_i, \mathcal{E}_j) \notin E\}$.

**Theorem 6.1.2** (Lovász Local Lemma)**.** *Let $\{\mathcal{E}_1, \ldots, \mathcal{E}_n\}$ be a set of events such that*

- *for $i \in \{1, \ldots, n\}$, $\mathbb{P}(\mathcal{E}_i) \leq p$,*
- *the degree of the dependency graph for $\{\mathcal{E}_1, \ldots, \mathcal{E}_n\}$ is at most $d$, and*
- *$4dp \leq 1$.*

*Then $\mathbb{P}(\bigwedge_{i=1}^n \overline{\mathcal{E}}_i) > 0$.*

To facilitate induction, we will prove a stronger lemma. Let's first define $F_S = \bigwedge_{i \in S} \overline{\mathcal{E}}_i$, for every $S \subseteq \{1, \ldots, n\}$. The Lovász Local Lemma follows from the first part of the following lemma, when $S = \{1, \ldots, n\}$.

**Lemma 6.1.3.** *For every $S \subseteq \{1, \ldots, n\}$:*

$$\mathbb{P}(F_S) > 0 \tag{6.1}$$

$$\forall i \in \{1, \ldots, n\} \setminus S,\ \mathbb{P}(\mathcal{E}_i \mid F_S) \leq 2p \tag{6.2}$$

*Proof.* The proof is by induction on the size of $S$. The base case is $|S| = 0$ (so $S = \emptyset$). This case is straightforward because $\mathbb{P}(F_S) = 1$ and $\mathbb{P}(\mathcal{E}_i \mid F_S) = \mathbb{P}(\mathcal{E}_i) \leq p \leq 2p$.

For the inductive step, to show the first property, fix some $t \in S$. Then

$$\begin{aligned}\mathbb{P}(F_S) &= \mathbb{P}(\overline{\mathcal{E}}_t \wedge F_{S\setminus\{t\}}) \\ &= \mathbb{P}(\overline{\mathcal{E}}_t \mid F_{S\setminus\{t\}})\mathbb{P}(F_{S\setminus\{t\}}) \\ &\geq (1-2p)\mathbb{P}(F_{S\setminus\{t\}}) \\ &> 0,\end{aligned}$$

by the induction hypothesis and the fact that $p < 1/2$, which follows because $4dp \leq 1$.

For the inductive step, to show the second property, let's partition $S$ into two sets — let $S_1$ be the set of events in $S$ adjacent to $\mathcal{E}_i$ and let $S_2 = S \setminus S_1$.

The case $S_1 = \emptyset$ is easy because in this case $\mathbb{P}(\mathcal{E}_i \mid F_S) = \mathbb{P}(\mathcal{E}_i) \leq p \leq 2p$. Otherwise, we have

$$\mathbb{P}(\mathcal{E}_i \mid F_S) = \frac{\mathbb{P}(\mathcal{E}_i \wedge F_S)}{\mathbb{P}(F_S)} = \frac{\mathbb{P}(\mathcal{E}_i \wedge F_{S_1} \mid F_{S_2})\mathbb{P}(F_{S_2})}{\mathbb{P}(F_{S_1} \mid F_{S_2})\mathbb{P}(F_{S_2})} = \frac{\mathbb{P}(\mathcal{E}_i \wedge F_{S_1} \mid F_{S_2})}{\mathbb{P}(F_{S_1} \mid F_{S_2})}.$$

Note that we used the first property of the lemma: $\mathbb{P}(F_S) > 0$.

We now bound the numerator and denominator separately. Using the premises of the lemma, we bound the numerator by

$$\mathbb{P}(\mathcal{E}_i \wedge F_{S_1} \mid F_{S_2}) \leq \mathbb{P}(\mathcal{E}_i \mid F_{S_2}) = \mathbb{P}(\mathcal{E}_i) \leq p.$$

Next, using the inductive hypothesis (since $|S_2| < |S|$) and the union bound, we bound the denominator by

$$\begin{aligned}\mathbb{P}(F_{S_1} \mid F_{S_2}) &= \mathbb{P}(\bigwedge_{j\in S_1} \overline{\mathcal{E}}_j \mid F_{S_2}) \\ &= 1 - \mathbb{P}(\cup_{j\in S_1}\mathcal{E}_j \mid F_{S_2}) \\ &\geq 1 - \sum_{j\in S_1} \mathbb{P}(\mathcal{E}_j \mid F_{S_2}) \\ &\geq 1 - \sum_{j\in S_1} 2p \\ &\geq 1 - 2pd \\ &\geq 1/2.\end{aligned}$$

□

Slightly stronger versions of the Lovász Local Lemma exist.