18.415/6.854 Advanced Algorithms	September 1994

Randomized Algorithms

Lecturer: Michel X. Goemans

1 Introduction

We have already seen some uses of randomization in the design of on-line algorithms. In these notes, we shall describe other important illustrations of randomized algorithms in other areas of the theory of algorithms. For those interested in learning more about randomized algorithms, we strongly recommend the forthcoming book by Motwani and Raghavan. [9]. First we shall describe some basic principles which typically underly the construction of randomized algorithms. The description follows a set of lectures given by R.M. Karp [7].

1. **Foiling the adversary**. This does not need much explanation since this was the main use of randomization in the context of on-line algorithms. It applies to problems which can be viewed as a game between the algorithm designer and the adversary. The adversary has some payoff function (running time of the algorithm, cost of solution produced by algorithm, competitive ratio, ...) and the algorithm designer tries to minimize the adversary's payoff. In this framework randomization helps in confusing the adversary. The adversary cannot predict the algorithm's moves.

2. **Abundance of witnesses**. Often problems are of the type "does input have property p?" (for example. "Is n composite?"). Typically the property of interest can be established by providing an object, called a "witness". In some cases, finding witnesses deterministically may be difficult. In such cases randomization may help. For example if there exists a probability space where witnesses are abundant then a randomized algorithm is likely to find one by repeated sampling. If repeated sampling yields a witness, we have a mathematical proof that the input has the property. Otherwise we have strong evidence that the input doesn't have the property, but no proof. A randomized algorithm which may return an incorrect answer is called a *Monte-Carlo* algorithm. This is in contrast with a *Las Vegas* algorithm which is guaranteed to return the correct answer.

3. **Checking an identity**
For example, given a function of several variables $f(x_1, \ldots, x_n)$, is $f(x_1, \ldots, x_n) \equiv 0$? One way to test this is to generate a random vector $a_1, \ldots, a_n$ and evaluate $f(a_1, \ldots, a_n)$. If its value is not 0, then clearly $f \not\equiv 0$.

Suppose we can generate random vectors $a_1, \ldots, a_n$ under some probability distribution so that

$$P = Pr\left[f(a_1, \ldots, a_n) = 0 | f \not\equiv 0\right] < \frac{1}{2},$$

or any other constant bounded away from 1. Then we can determine whether or not $f \equiv 0$ with high probability. Notice that this is a special case of category 2, since in this probability space, vectors a for which $f(a_1, \ldots, a_n) \neq 0$ constitute "witnesses".

4. **Random ordering of input**
   The performance of an algorithm may depend upon the ordering of input data; using randomization this dependence is removed. The classic example is Quicksort, which takes $O(n^2)$ time in the worst case but when randomized takes $O(n \lg n)$ expected time, and the running time depends only on the coin tosses, not on the input. This can be viewed as a special case of category 1. Notice that randomized quicksort is a Las Vegas algorithm; the output is always correctly sorted.

5. **Fingerprinting**
   This is a technique for representing a large object by a small fingerprint. Under appropriate circumstances, if two objects have the same fingerprint, then there is strong evidence that they are identical.

   An example is the randomized algorithm for pattern matching by Karp and Rabin [8]. Suppose we are given a string of length n such as

   $$randomizearandomlyrandomrandomizedrandom$$

   and a pattern of size m such as *random*. The task is to find all the places the pattern appears in the long string.

   Let us first describe our model of computation. We assume a simplified version of the unit-cost RAM model in which the standard operations $+, -, *, /, <, =$ take one unit of time provided they are performed over a field whose size is polynomial in the input size. In our case, the input size is $O(n+m) = O(n)$ and thus operations on numbers with $O(\log n)$ bits take only one unit of time.

   A naive approach is to try starting at each location and compare the pattern to the m characters starting at that location; this takes $O(nm)$ time (in our model of computation we cannot compare two strings of m characters in $O(1)$ time unless $m = O(\log n)$). The best deterministic algorithm takes $O(n+m)$ time, but it is complicated. There is, however, a fairly simple $O(n+m)$ time randomized algorithm.

   Say that the pattern X is a string of bits $x_1, \ldots, x_m$ and similarly $Y = y_1, \ldots, y_n$. We want to compare X, viewed as a number to $Y_i = y_i, \ldots, y_{i+m-1}$. This would normally take $O(m)$ time, but it can be done much more quickly by computing fingerprints and comparing those. To compute fingerprints, choose a prime p. Then the fingerprint of X is $h(X) = X \bmod p$ and similarly $h(Y_i) = Y_i \bmod p$. Clearly $h(X) \neq h(Y_i) \Rightarrow X \neq Y_i$. The converse is not necessarily true. Say that we have a *false match* if $h(X) = h(Y_i)$ but $X \neq Y_i$. A false match occurs iff p divides $|X - Y_i|$.

We show that if p is selected uniformly among all primes less than some threshold Q then the probability of a small match is small. First how many primes p divide $|X - Y_i|$? Well, since every prime is at least 2 and $|X - Y_i| \leq 2^m$, we must have at most m primes dividing $|X - Y_i|$. As a result, if p is chosen uniformly at random among $\{q : q \text{ prime and } q \leq Q\}$ then $Pr[h(X) = h(Y_i) | X \neq Y_i] \leq \frac{m}{\pi(Q)}$, where $\pi(n)$ denotes the number of primes less or equal to n. Thus, the probability that there is a false match for some i is upper bounded by n times $\frac{m}{\pi(Q)}$. Since $\pi(n)$ is asymptotically equal to $n/\ln n$, we derive that this probability is $O(\frac{\ln n}{n})$ if $Q = n^2 m$. This result can be refined by using the following lemma and the fact that there is a false match for some i if p divides $\prod_i |X - Y_i| \leq 2^{nm}$.

Lemma 1 *The number of primes dividing $a \leq 2^n$ is at most $\pi(n) + O(1)$.*

The refined version is the following:

Theorem 2 *If p is chosen uniformly at random among $\{q : q \text{ prime and } q \leq n^2 m\}$, then the probability of a false match for some i is upper bounded by $\frac{2+O(1)}{n}$.*

The fingerprint has only $\lg(n^2 m)$ bits, much smaller than m. Operations on the fingerprints can be done in $O(1)$ time in our computational model.

The advantage of this approach is that it is easy to compute $h(Y_{i+1})$ from $h(Y_i)$ in $O(1)$ time:

$$\begin{aligned} Y_{i+1} &= 2Y_i + y_{i+m} - 2^m y_i \\ h(Y_{i+1}) &= 2h(Y_i) + y_{i+m} - 2^m y_i \pmod{p}. \end{aligned}$$

One then checks if the fingerprints are equal. If they are, the algorithm claims that a match has been found and continues. To reduce the probability of failure, one can repeat the algorithm with another prime (or several other primes) and ouput only those who were matches for all primes tried. This is thus a Monte Carlo algorithm whose running time is $O(n + m)$.

This algorithm can easily be transformed into a Las Vegas algorithm. Whenever there is a potential match (i.e. the fingerprints are equal), we compare X and Y_i directly at a cost of $O(m)$. The *expected* running time is now $O((n + m) + km + nm\frac{2}{n}) = O(km + n)$, where k denotes the number of real matches.

6. **Symmetry breaking**
   This is useful in distributed algorithms, but we won't have much to say about it in this class. In that context, it is often necessary for several processors to collectively decide on an action among several (seemingly indistinguishable) actions, and randomization helps in this case.

7. **Rapidly mixing Markov chains**
   These are useful for counting problems, such as counting the number of cycles in a graph, or the number of trees, or matchings, or whatever. First, the counting problem is transformed into a sampling problem. Markov chains can be used to generate points of a given space at random, but we need them to converge rapidly — such Markov chains are called rapidly mixing. This area is covered in details in these notes.

2 Randomized Algorithm for Bipartite Matching

We now look at a randomized algorithm by Mulmuley, Vazirani and Vazirani [10] for bipartite matching. This algorithm uses randomness to check an identity.

Call an undirected graph $G = (V, E)$ bipartite if (1) $V = A \cup B$ and $A \cap B = \emptyset$, and (2) for all $(u, v) \in E$, either $u \in A$ and $v \in B$, or $u \in B$ and $v \in A$. An example of a bipartite graph is shown in Figure 1.

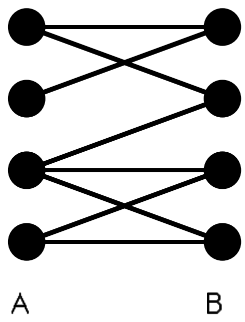

Figure 1: Sample bipartite graph.

A *matching* on G is a collection of vertex-disjoint edges. A *perfect matching* is a matching that covers every vertex. Notice that we must have $|A| = |B|$.

We can now pose two problems:

1. Does G have a perfect matching?

2. Find a perfect matching or argue that none exists.

Both of these problems can be solved in polynomial time. In this lecture we show how to solve the first problem in randomized polynomial time, and next lecture we'll cover the second problem. These algorithms are simpler than the deterministic ones, and lead to parallel algorithms which show the problems are in the class RNC. RNC is Randomized NC, and NC is the complexity class of problems that can be solved in polylogarithmic time on a number of processes polynomial in the size of the input. No NC algorithm for either of these problems is known.

The Mulmuley, Vazirani and Vazirani randomized algorithm works as follows. Consider the adjacency matrix A on graph $G = (V, E)$ whose entries a_{ij} are defined as follows:

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

where the indices i and j correspond to vertices of the vertex sets A and B respectively.

There exists a perfect matching in the graph G if and only if the adjacency matrix contains a set of n 1's, no two of which are in the same column or row. In other words, if all other entries were 0 a permutation matrix would result.

Consider the function called the *permanent* of A, defined as follows:

$$perm(A) = \sum_{permutations\,\sigma} \left(\prod_{i=1}^{n} a_{i\sigma_i} \right). \tag{2}$$

This gives the number of perfect matchings of the graph A represents. Unfortunately the best known algorithm for computing the permanent has running time $O(n2^n)$. However, note the similarity of the formula for computing the permanent to that for the determinant of A:

$$det(A) = \sum_{permutations\,\sigma} sign(\sigma) \left(\prod_{i=1}^{n} a_{i\sigma_i} \right).$$

The determinant can be computed in $O(n^3)$ time by using Gaussian elimination (and in $O(\log^2(n))$ time on $O(n^{3.5})$ processors). Note also that:

$$det(A) \neq 0 \Rightarrow perm(A) \neq 0 \Leftrightarrow \exists \text{ perfect matching.}$$

Unfortunately the converse is not true.

To handle the converse, we replace each entry a_{ij} of matrix A with $(a_{ij} x_{ij})$, where x_{ij} is a variable. Now both $det(A)$ and $perm(A)$ are polynomials in x_{ij}. It follows

$$det(A) \equiv 0 \Leftrightarrow perm(A) \equiv 0 \Leftrightarrow \nexists \text{ perfect matching.}$$

A polynomial in 1 variable of degree n will be identically equal to 0 if and only if it is equal to 0 at $n+1$ points. So, if there was only one variable, we could compute this determinant for $n+1$ values and check whether it is identically zero. Unfortunately, we are dealing here with polynomials in several variables.

So to test whether $det(A) \equiv 0$, we will generate values for the x_{ij} and check if the resulting matrix has $det(A) = 0$ using Gaussian elimination. If it is not 0, we know the determinant is not equivalent to 0, so G has a perfect matching.

Theorem 3 *Let the values of the x_{ij} be independently and uniformly distributed in $[1, 2, \ldots, 2n]$, where $n = |A| = |B|$. Let A' be the resulting matrix. Then*

$$Pr[det(A') = 0 | det(A) \not\equiv 0] \leq \frac{1}{2}.$$

It follows from the theorem that if G has a perfect matching, we'll find a witness in k trials with probability at least $1 - 1/2^k$.

In fact, this theorem is just a statement about polynomials. We can restate it as follows:

Theorem 4 *Let $f(x_1, \ldots, x_n)$ be a multivariate polynomial of degree d. Let x_i be independently and uniformly distributed in $\{1, 2, \ldots, 2d\}$. Then*

$$Pr[f(x_1, \ldots, x_n) \neq 0 | f \not\equiv 0] \geq \frac{1}{2}.$$

This theorem can be used for other problems as well.

Instead of proving this theorem we'll prove a stronger version which can be used for the second problem, that of finding the perfect matching.

Consider assigning costs c_{ij} ($c_{ij} \in \mathbb{N}$) to the edges $(i, j) \in E$. Define the cost of a matching M as:

$$c(M) = \sum_{(i,j) \in M} c_{ij}.$$

Now, consider the matrix A with entries $a_{ij} w_{ij}$, where $w_{ij} = 2^{c_{ij}}$. Then:

$$perm(A) = \sum_M 2^{c(M)}$$

and

$$det(A) = \sum_M sign(M) 2^{c(M)}.$$

If a unique minimum cost matching with cost c^* exists then $\det(A)$ will be nonzero and, in fact, it will be an odd multiple of 2^{c^*}.

We will prove that if we select the costs according to a suitable probability distribution then, with high probability, there exists a unique minimum cost matching. Let c_{ij} be independent, identically distributed random variables with distribution uniform in the interval $[1, \ldots, 2m]$, where $m = |E|$. The algorithm computes $\det(A)$ and claims that there is a perfect matching if and only if $\det(A)$ is nonzero. The only situation in which this algorithm can err is when there is a perfect matching, but $\det(A) = 0$. This is thus a Monte-Carlo algorithm. The next theorem upper-bounds the probability of making a mistake.

Theorem 5 *Assume that there exists a perfect matching in G. Then the probability that we will err with our algorithm is at most $\frac{1}{2}$.*

If a higher reliability is desired then it can be attained by running multiple passes, and only concluding that there is no perfect matching if no pass can find one.

Proof:

We need to compute $Pr\left[det(A)=0\right]$. Though this quantity is difficult to compute, we can fairly easily find an upper bound for it. As we have seen previously,

$$\begin{aligned} Pr\left[det(A)=0\right] &= 1 - Pr\left[det(A)\neq 0\right] \\ &\leq 1-P \end{aligned}$$

where

$$P = Pr\left[\exists \text{ unique minimum cost matching}\right].$$

Indeed, if there is a unique minimum cost matching of cost say c^* then $det(A)$ is an odd multiple of 2^{c^*} and, hence, non-zero. The following claim completes the proof.

Claim 6 $P \geq \frac{1}{2}$

Given a vector c, define d_{ij} to be the maximum value for c_{ij} such that (i,j) is part of some minimum cost matching.

We can then draw the following inferences:

$$\begin{cases} c_{ij} > d_{ij} \Rightarrow (i,j) \textit{ is not part of ANY minimum cost matching} \\ c_{ij} = d_{ij} \Rightarrow (i,j) \textit{ is part of SOME minimum cost matching} \\ c_{ij} < d_{ij} \Rightarrow (i,j) \textit{ is part of EVERY minimum cost matching} \end{cases}$$

Thus, if $c_{ij} \neq d_{ij}$ for all $(i,j) \Rightarrow \exists$ *a unique minimum cost matching* M. Moreover, this matching is given by $M = \{(i,j) \mid c_{ij} < d_{ij}\}$.

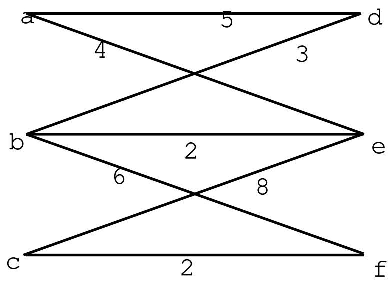

Figure 2: Example graph for d_{ij} computations.

Figure 3 shows an example of a bipartite graph with the values of c_{ij} assigned. Notice that in this graph $c^* = 9$. Consider the edge (c,f). The cheapest perfect matching not containing (c,f) has a cost of $6+8+5=19$. The other edges in the perfect matching containing (c,f) have a total cost of 7, so $d_{cf} = 12$. Thus, it is in every perfect matching. $d_{ad} = 5 = 4+3+2-2-2 = c_{ad}$. Thus it is in some minimum cost matching. Finally, $d_{ce} = -2 = 9-6-5 < c_{ce}$, so (c, e) is not in any minimum cost matching.

Therefore,

$$\begin{aligned} Pr[\mathit{unique\ minimum\ cost\ matching}] &\geq Pr\left[c_{ij} \neq d_{ij} \text{ for all } (i,j)\right] \\ &= 1 - Pr\left[c_{ij} = d_{ij} \; for\ some\ (i,j)\right] \\ &\geq 1 - \sum_{(i,j)\in E} Pr\left[c_{ij} = d_{ij}\right] \\ &\geq 1 - m \cdot \frac{1}{2m} \\ &= \frac{1}{2}. \end{aligned}$$

The equation in the next to last line is justified by our selection of $m = |E|$ and the fact that d_{ij} is independent of c_{ij}, so that the probability of c_{ij} being equal to the particular value d_{ij} is either $\frac{1}{2m}$ iff d_{ij} is in the range $[1, \ldots, 2m]$ or 0 otherwise. $\diamond$

$\square$

Notice that if we repeat the algorithm with new random c_{ij}'s, then the second trial will be independent of the first run. Thus, we can arbitrarily reduce the error probability of our algorithm, since the probability of error after t iterations is at most $\left(\frac{1}{2}\right)^t$.

Also, note that, in the proof, we do not make use of the assumption that we are working with matchings. Thus, this proof technique is applicable to a wide class of problems.

2.1 Constructing a Perfect Matching

In order to construct a perfect matching, assume that there exists a unique minimum cost matching (which we have shown to be true with probability at least $\frac{1}{2}$) with cost c^*. The determinant of A will then be an odd multiple of 2^{c^*}. By expanding the determinant along row i, it can be computed by the following formula:

$$\sum_j \pm 2^{c_{ij}} a_{ij} det(A_{ij}) \tag{3}$$

where A_{ij} is the matrix created by removing column i and row j from the matrix A (See figure 3), and the sign depends on the parity of $i+j$. The term in the summation above will be an odd multiple of 2^{c^*} if $(i, j) \in M$ and an even multiple otherwise. So, we can reconstruct a perfect matching M by letting:

$$M = \left\{(i,j) \mid 2^{c_{ij}} det(A_{ij}) \; is\ an\ odd\ multiple\ of\ 2^{c^*}\right\}. \tag{4}$$

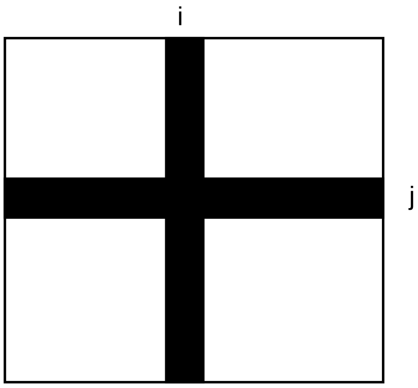

Figure 3: The Matrix A_{ij}. The matrix A_{ij} is formed by removing the ith column and the jth row from the matrix A.

Note that c^* can be obtained since 2^{c^*} is the largest power of 2 in $det(A)$.

The algorithm we have presented can be seen to be in RNC since a determinant can be computed in RNC.

We can apply a similar algorithm for solving the following related matching problem:

> Given:
>
> - A bipartite graph G,
> - A coloring for each edge in G of either *red* or *blue*,
> - an integer k,
>
> find a perfect matching with exactly k red edges.

However, in contrast to the problem of finding any perfect matching, it is not known whether this problem is in P or even NP-complete. For this problem, define the entries a_{ij} of the matrix A as follows:

$$a_{ij} = \begin{cases} 0 & \text{if } (i,j) \notin E \\ w_{ij} & \text{if } (i,j)\ is\ blue \\ w_{ij}x & \text{if } (i,j)\ is\ red \end{cases} \tag{5}$$

where x is a variable. Both the permanent of A and the determinant of A are now polynomials in one variable, x, and we wish to know the coefficients c_k of x^k. If all w_{ij} were 1, c_k would represent the number of perfect matchings with exactly k red edges. If there does exist a perfect matching with k red eges, then $\Pr(c_k = 0) \leq \frac{1}{2}$ by the same argument we derived the probability that $\det(A) = 0$ when a perfect matching exists, since we can always decompose the determinant into a sum of products of matrices with x^k.

We can now compute all the c_k by computing the determinant of A in $n+1$ different points and interpolating from that data to compute the coefficients.

3 Markov Chains

A lot of recent randomized algorithms are based on the idea of rapidly mixing Markov chains. A Markov chain is a stochastic process, i.e. a random process that evolves with time. It is defined by:

- A set of states (that we shall assume to be finite) $1, \ldots, N$.
- A transition matrix P where the entry p_{ij} represents the probability of moving to state j when at state i, i.e. $p_{ij} = Pr\left[X_{t+1} = j \mid X_t = i\right]$, where X_t is a random variable denoting the state we are in at time t.

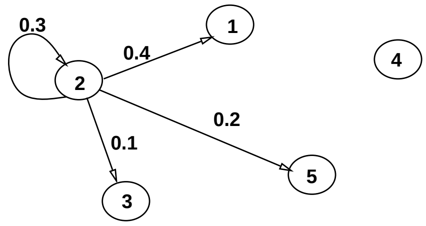

Figure 4: A Markov Chain.

Figure 4 partially illustrates a set of states having the following transition matrix:

$$P = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 \\ 0.4 & 0.3 & 0.1 & 0 & 0.2 \\ 0 & 0.5 & 0 & 0 & 0.5 \\ 0.2 & 0.8 & 0 & 0 & 0 \\ 0.1 & 0.1 & 0.1 & 0.1 & 0.6 \end{pmatrix} \tag{6}$$

The transition matrix P satisfies the following two conditions and any such matrix is called "stochastic":

- $P \geq 0$
- $\sum\limits_j p_{ij} = 1$ for all i.

Suppose we are given an initial probability distribution $\pi^{(0)}$ where we denote $\Pr(X_0 = i)$ by $\pi_i^{(0)}$. Then,

$$\begin{aligned}
\pi_j^{(1)} &= Pr(X_1 = j) \\
&= \sum_i Pr(X_1 = j \mid X_0 = i) Pr(X_0 = i) \\
&= \sum_{i=1}^{n} p_{ij} \cdot \pi_i^0,
\end{aligned}$$

i.e.

$$\pi^{(1)} = \pi^{(0)} \cdot P.$$

Repeating this argument, we obtain

$$\pi^{(s)} = \pi^{(0)} \cdot P^s.$$

Here, $\pi^{(s)}$ represents the distribution after s steps. Therefore, the matrix P^s is the so-called "s-step transition matrix". Its entries are $p_{ij}^s = Pr\left[X_{t+s} = j \mid X_t = i\right]$.

Definition 1 *A Markov Chain is said to be "ergodic" if* $\lim_{s\to\infty} p_{ij}^s = \pi_j > 0$ *for all* j *and is independent of* i.

In this case,

$$\begin{aligned}
P^\infty &= \lim_{s\to\infty} \left[P^s\right] \\
&= \begin{bmatrix} \pi_1 & \dots & \pi_j & \dots & \pi_n \\ \vdots & & \vdots & & \vdots \\ \pi_1 & \dots & \pi_j & \dots & \pi_n \end{bmatrix}
\end{aligned}$$

Hence, π is independent of the starting distribution $\pi^{(0)}$:

$$\pi = \pi^{(0)} \cdot P^\infty.$$

Any vector π which satisfies $\pi P = \pi$ and $\sum_i \pi_i = 1$ is called a *stationary distribution.*

Proposition 7 *For an ergodic MC,* π *is a stationary distribution, and moreover it is the unique stationary distribution.*

Proof:

We have already shown that $\pi^{(0)} P = \pi^{(1)}$ which implies

$$P^\infty = \lim_{s\to\infty} P^{s+1} = (\lim_{s\to\infty} P^s) P = P^\infty \ P$$

Since $\pi^{(0)}P^\infty = \pi$ for any probability distribution $\pi^{(0)}$, we have $\pi^{(0)}P^\infty = \pi^{(0)}P^\infty P$ which implies $\pi P = \pi$. Since $P \cdot 1 = 1$ where 1 is the vector of 1's, we derive that $P^\infty \cdot 1 = 1$, which says that $\sum_i \pi_i = 1$.

The reason why there is a unique stationary distribution is simply because by starting from another stationary distribution, say $\tilde{\pi}$, we always remain in this distribution implying that $\tilde{\pi} = \pi$. □

Proposition 7 gives an "easy" way to calculate the limiting distribution of an ergodic Markov chain from the transition matrix P. We just solve the linear system of equations $\pi P = \pi$, $\sum_i \pi_i = 1$.

4 Ergodicity and time reversibility

Theorem 8 *An MC is ergodic if and only if both of the following are true:*

1. *it is* irreducible. *That is, the underlying graph (consisting of states and transitions with positive probabilities on them) is strongly connected. Formally, for all i and j there is s such that $p_{ij}^{(s)} > 0$.*

2. *the chain is* aperiodic. *That is, you cannot divide states into subgroups so that you must go from one to another in succession. Formally,*

$$\gcd\{s : p_{ij}^{(s)} > 0\} = 1$$

for all i and j.

Definition 2 *An ergodic MC is called* **time reversible** *(TR) if the chain remains a Markov chain when you "go backwards". More formally, if π is the stationary distribution, then*

$$\pi_i p_{ij} = \pi_j p_{ji}$$

for all pairs of states i and j or, in words, the expected (or ergodic) flow from i to j equals the expected flow from j to i.

Proposition 9 *Consider an ergodic MC. Suppose there exists γ such that the* balance *conditions are satisfied: $\gamma_i p_{ij} = \gamma_j p_{ji}, \forall i, j$ and also, $\sum_i \gamma_i = 1$. Then γ is the stationary distribution, and clearly, the MC is TR.*

Clearly the MC in Figure 5 is ergodic (strong connectivity (i.e., irreducibility) and aperiodicity are obvious). It is clear that there exists a stationary distribution, and we can easily guess one. Consider $\pi_1 = \frac{1}{3}$ and $\pi_2 = \frac{2}{3}$. Since one can easily verify that π satisfies the balance conditions, π must be the stationary distribution (and the MC is time-reversible).

Consider an ergodic MC which is also symmetric ($p_{ij} = p_{ji}$) as in Figure 6. Then the stationary distribution is $\pi_i = \frac{1}{N}$, where N is the number of states.

In these notes, we shall consider MC's that are ergodic and symmetric, and therefore, have a uniform stationary distribution over states.

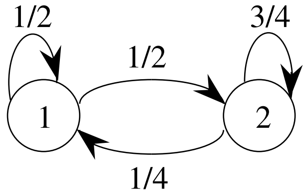

Figure 5: An example of an MC with a stationary distribution.

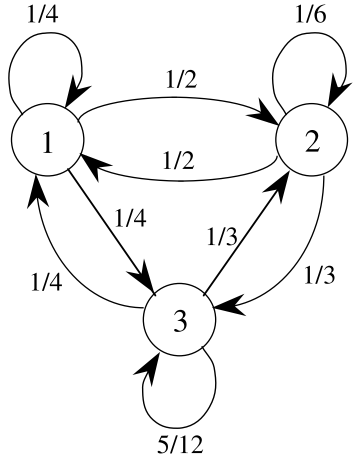

Figure 6: A symmetric ergodic MC.

5 Counting problems

Suppose you would like to count the number of blue-eyed people on a planet. One approach is to take some sort of census, checking everyone on the planet for blue eyes. Usually this is not feasible. If you know the total number of people on the planet, then you could take a sample in such a way that every person has the same probability of being selected. This would be a uniformly selected sample of size n out of a universe of size N. Let Y be the random variable representing the number of individuals in this sample that have the property (blue eyes, in our example). Then we can infer that the total number of individuals with this property is approximately $\frac{YN}{n}$.

If the actual number of individuals with the property is pN, then we have a Bernoulli process with n trials and success probability p. The random variable Y has a Binomial distribution with parameters (n, p). We are interested in finding out how close $\frac{Y}{n}$ is to p, and to do this we can use *Chernoff bounds*, which are exponentially decreasing on the tail of the distribution. Since Chernoff bounds are quite useful, we will digress for a while and derive them in some generality.

Lemma 10 *Let X_i be independent Bernoulli random variables with probability of success p_i. Then, for all $\alpha > 0$ and all $t > 0$, we have*

$$Pr\left[\sum_{i=1}^{n} X_i > t\right] \leq e^{-\alpha t} \prod_{i=1}^{n} E\left[e^{\alpha X_i}\right] = e^{-\alpha t} \prod_{i=1}^{n} \left(p_i e^{\alpha} + (1 - p_i)\right).$$

Proof:

$$Pr\left[\sum_{i=1}^{n} X_i > t\right] = Pr\left[e^{\alpha \sum_{i=1}^{n} X_i} > e^{\alpha t}\right]$$

for any $\alpha > 0$. Moreover, this can be written as $Pr[Y > a]$ with $Y \geq 0$. From Markov's inequality we have

$$Pr[Y > a] \leq \frac{E[Y]}{a}$$

for any nonnegative random variable. Thus,

$$\begin{aligned} Pr\left[\textstyle\sum_{i=1}^{n} X_i > t\right] &\leq e^{-\alpha t} E\left[e^{\alpha \sum_i X_i}\right] \\ &= e^{-\alpha t} \textstyle\prod_{i=1}^{n} E\left[e^{\alpha X_i}\right] \text{ because of independence.} \end{aligned}$$

The equality then follows from the definition of expectation. □

Setting $t = (1 + \epsilon)E[\sum_i X_i]$ for some $\epsilon > 0$ and $\alpha = \ln(1 + \epsilon)$, we obtain:

Corollary 11 *Let X_i be independent Bernoulli random variables with probability of success p_i, and let $np = E[\sum_{i=1}^{n} X_i] = \sum_{i=1}^{n} p_i$. Then, for all $\epsilon > 0$, we have*

$$Pr\left[\sum_{i=1}^{n} X_i > (1 + \epsilon)np\right] \leq (1 + \epsilon)^{-(1+\epsilon)np} \prod_{i=1}^{n} E\left[(1 + \epsilon)^{X_i}\right] \leq \left[\frac{e^{\epsilon}}{(1 + \epsilon)^{(1+\epsilon)}}\right]^{np}.$$

The second inequality of the corollary follows from the fact that

$$E\left[(1+\epsilon)^{X_i}\right] = p_i(1+\epsilon) + (1-p_i) = 1 + \epsilon p_i \le e^{\epsilon p_i}.$$

For ϵ in the range $(0,1)$, we can simplify the above expression and write the following more classical form of the Chernoff bound:

Theorem 12 (Chernoff bound) *Let X_i be independent Bernoulli random variables with probability of success p_i, let $Y = \sum_{i=1}^n X_i$, and let $np = \sum_{i=1}^n p_i$. Then, for $1 > \epsilon > 0$, we have*

$$Pr\left[Y - np > \epsilon np\right] \le e^{-\epsilon^2 np/3}.$$

(For other ranges of ϵ, we simply have to change the constant $1/3$ appropriately in the exponent.)

Similarly, we can write a Chernoff bound for the probability that Y is below the mean.

Theorem 13 (Chernoff bound) *Let X_i be independent Bernoulli random variables with probability of success p_i, let $Y = \sum_{i=1}^n X_i$, and let $np = \sum_{i=1}^n p_i$. Then, for $1 > \epsilon > 0$, we have*

$$Pr\left[Y - np < \epsilon np\right] \le \left[\frac{e^{\epsilon}}{(1-\epsilon)^{(1-\epsilon)}}\right]^{np} \le e^{-\epsilon^2 np/2}.$$

The last upper bound of $e^{-\epsilon^2/2}$ can be derived by a series expansion.

Let us go back to our counting problem. We can use Theorems 12 and 13 to see what sample size n we need to ensure that the relative error in our estimate of p is arbitrarily small. Suppose we wish to impose the bound $\Pr\left\{|\frac{Y}{n} - p| > \epsilon p\right\} \le \delta$. Imposing $e^{-\epsilon^2 np/3} \le \frac{\delta}{2}$, we derive that we can let the number of samples to be

$$n = \frac{3}{\epsilon^2 p} \log \frac{2}{\delta}.$$

Notice that n is polynomial in $\frac{1}{\epsilon}$, $\log \frac{1}{\delta}$, and $\frac{1}{p}$. If p is exponentially small, then this may be a bad approach. For example, if we were trying to count the number of American citizens who have dodged the draft, have become President of the country and who are being sued for sexual harassment, we would need an exponential number of trials.

These notions can be formalized as follows. Suppose we would like to compute an integral number $f(x)$ (x represents the input).

Definition 3 *An fpras (fully polynomial randomized approximation scheme) for $f(x)$ is a randomized algorithm which given x and ϵ outputs an integer $g(x)$ such that*

$$Pr\left[\left|\frac{g(x) - f(x)}{f(x)}\right| \le \epsilon\right] \ge \frac{3}{4}$$

and runs in time polynomial in the size of the input x and in $\frac{1}{\epsilon}$.

Thus repeated sampling can be used to obtain an fpras if we can view $f(x)$ as the number of elements with some property in a universe which is only polynomially bigger, and if we can sample uniformly from this universe. Notice that the running time is assumed to be polynomial in $\frac{1}{\epsilon}$. If we were to impose the stronger condition that the running time be polynomial in $\ln \frac{1}{\epsilon}$ then we would be able to compute $f(x)$ exactly in randomized polynomial time whenever the size of the universe is exponential in the input size (simply run the fpras with ϵ equal to the inverse of the size of the universe).

Going back to our original counting problem and assuming that p is not too small, the question now is how to draw a uniformly selected individual on the planet (or more generally a uniformly generated element in the universe under consideration). One possible approach is to use a Markov chain where there is one state for each individual. Assuming that each individual has at most 1000 friends, and that "friendship" is symmetric, we set the transition probability from an individual to each of his friends to be $\frac{1}{2000}$. Then if an individual has k friends, the transition probability to himself will be $1 - \frac{k}{2000} \geq \frac{1}{2}$, implying that the chain is aperiodic.

If the graph of friendship is strongly connected (everyone knows everyone else through some sequence of friends of friends) then this MC is ergodic, and the stationary distribution is the uniform distribution on states.

Recall that

$$\lim_{s \to \infty} P^s = P^\infty = \begin{pmatrix} \pi_1 & \cdots & \pi_j & \cdots & \pi_n \\ \pi_1 & \cdots & \pi_j & \cdots & \pi_n \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \pi_1 & \cdots & \pi_j & \cdots & \pi_n \end{pmatrix}$$

If this limit converges quickly, we can simulate the MC for a finite number of steps and get "close" to the stationary distribution. Therefore, it would be useful for us to know the rate of convergence to the stationary distribution if we want to use Markov chains to approximately sample for a distribution.

It turns out that the rate of convergence is related to the eigenvalues of the transition matrix, P. Given a stochastic matrix P (recall that P is stochastic if it is nonnegative and all row sums are 1) with eigenvalues $\lambda_1, \dots, \lambda_N$, we have the following.

1. All $|\lambda_i| \leq 1$. Indeed, if $Pe_i = \lambda e_i$ then $P^s e_i = \lambda^s e_i$, and the fact that the LHS is bounded implies that the RHS must also be.

2. Since P is stochastic, $\lambda_1 = 1$ ($P1 = 1$).

3. The MC is ergodic iff $|\lambda_i| < 1$ for all $i \neq 1$.

4. If P is symmetric then all eigenvalues are real.

5. if P is symmetric and stochastic then if $p_{ii} \geq \frac{1}{2}$ for all i then $\lambda_i \geq 0$ for all i. Indeed, if we let $Q = 2P - I$, then Q is stochastic. Hence, the ith eigenvalue for Q, $\lambda_i^Q = 2\lambda_i - 1$, but $|2\lambda_i - 1| \leq 1$ implies that $0 \leq \lambda_i \leq 1$.

6. **Speed of convergence**
   The speed of convergence is dictated by the second largest eigenvalue. For simplicity, suppose P has N linearly independent eigenvectors. Then P can be expressed as $A^{-1}DA$ where D is the diagonal matrix of eigenvalues ($D_{ii} = \lambda_i$). And
   $$P^2 = A^{-1}DAA^{-1}DA = A^{-1}D^2A,$$
   or, in general,
   $$\begin{aligned} P^s &= A^{-1}D^sA \\ &= \sum_{i=1}^{N} \lambda_i^s M_i \\ &= M_1 + \sum_{i=2}^{N} \lambda_i^s M_i \end{aligned}$$
   where M_i is the matrix formed by regrouping corresponding terms from the matrix multiplication. If the MC is ergodic, then for $i \neq 1$, $\lambda_i < 1$, so $\lim_{s\to\infty} \lambda_i^s = 0$ implying $M_1 = P^\infty$. Then $P^s - P^\infty = \sum_{i=2}^{N} \lambda_i^s M_i$ is dominated by the term corresponding to $\lambda_{max} = \max_{i\neq 1} |\lambda_i|$. More generally,

Theorem 14 *Consider an ergodic time-reversible MC with stationary distribution π. Then the relative error after t steps is*

$$\Delta = \max_{i,j} \frac{|p_{ij}^{(t)} - \pi_j|}{\pi_j} \leq \frac{\lambda_{max}^t}{\min_j \pi_j}.$$

In particular, for an ergodic symmetric chain with $p_{ii} \geq \frac{1}{2}$, we have $\lambda_1 > \lambda_2 \geq \ldots \geq \lambda_N \geq 0$, so $\lambda_{max} = \lambda_2$.

Corollary 15 *For an ergodic symmetric MC with $p_{ii} \geq \frac{1}{2}$, the relative error $\Delta \leq \epsilon$ if $t \geq \frac{\log(N/\epsilon)}{\log(1/\lambda_2)}$.*

Returning to our example: In order to calculate how many iterations are needed until we are arbitrarily close to the uniform distribution π, we need to evaluate the second eigenvalue. For this purpose, Jerrum and Sinclair [11] have derived a relationship between the so-called *conductance* of the MC and λ_{max}. Their result can be viewed as an isoperimetric inequality, and its derivation is analogous to an isoperimetric inequality of Cheeger [4] for Riemannian manifolds, or results on expanders by Alon [1] and Alon and Milman [2].

6 Conductance of Markov chains (Jerrum-Sinclair)

Given a set S of states, let C_S denote the capacity of S, which is the probability of being in some state in S when steady-state is reached. Specifically,

$$C_S = \sum_{i \in S} \pi_i.$$

Define F_S, the ergodic flow out of S (expected flow) so that

$$F_S = \sum_{i \in S, j \notin S} \pi_i p_{ij},$$

(summing over transitions from S to the complement of S, $\overline{S}$). Clearly $F_S \leq C_S$.

Define $\Phi_S = F_S / C_S$, which is the probability of leaving S given that we are already in S. Define the *conductance* of the MC

$$\Phi := \min_{S : C_S \leq \frac{1}{2}} \Phi_S.$$

Intuitively, if we have an MC with small conductance, then once we are in a set S, we are "stuck" in S for a long time. This implies that it will take a long time to reach the stationary distribution, so the rate of convergence will be small. We might therefore expect that λ_2 will be close to 1 if the conductance is small.

Theorem 16 (Jerrum-Sinclair[11]) *For an ergodic MC that is TR, we have*

$$\lambda_2 < 1 - \Phi^2/2.$$

Remark 1 *There exist corresponding lower bounds expressing that $\Delta \geq \lambda_2^t$ and $\lambda_2 \geq 1 - 2\Phi$. This therefore shows that the conductance is an appropriate measure of the speed of convergence.*

7 Evaluation of Conductance of Markov Chains

Given a markov chain, the task will be to evaluate the conductance Φ. In order to generate Markov chains with a uniform steady state distribution and rapidly mixing property, we restrict our attention to the following MC: symmetric and equal transition probability p between states having a transition (i.e. for all $i \neq j$, either $p_{ij} = p_{ji} = 0$ or $p_{ij} = p_{ji} = p$). Therefore, it will have a uniform steady state distribution $\pi_i = 1/N$, where N denotes the number of states. Instead of looking at the MC, we can look at the underlying graph $G = (V, E)$, $E = \{(i,j) : p_{ij} = p_{ji} = p\}$. For a set S of states, let $\delta(S) = \{(i,j) \in E : i \in S, j \notin S\}$, then,

$$C_S = \sum_{i \in S} \pi_i = \frac{1}{N}|S|,$$

$$F_S = \sum_{i \in S, j \notin S} \pi_i p_{ij} = p \cdot \frac{1}{N} |\delta(S)|,$$

$$\Phi_S = \frac{F_S}{C_S} = p \frac{|\delta(S)|}{|S|}.$$

Definition 4 *The magnification factor of G is*

$$\gamma(G) = \min_{0 < |S| \leq \frac{|V|}{2}} \frac{|\delta(S)|}{|S|}.$$

Therefore,

$$\Phi = \min_S \Phi_S = p \cdot \gamma(G).$$

In the rest of this section, we study the conductance of a simple MC, which will be useful in the problem of counting matchings. Take a MC on states that are all binary numbers with d bits so that the underlying graph is a d-cube (with 2^d states). Two nodes are adjacent only if they differ by exactly one bit. Refer to Figure 7 for a 3-cube.

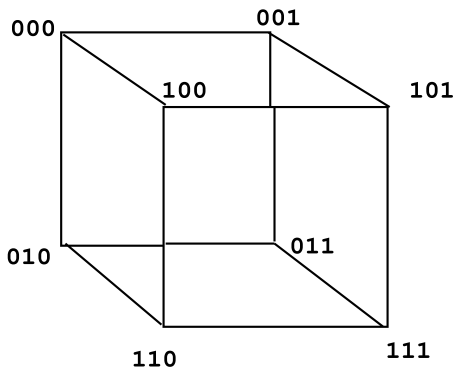

Figure 7: A 3-cube Markov chain.

If we put probabilities of $\frac{1}{2d}$ on all out-transitions then the self loops get probabilities $\frac{1}{2}$. The MC is symmetric and ergodic, so $\pi_i = \frac{1}{2^d}$. If we simulate a random walk on this d-cube, we obtain a random d-bit number.

Claim 17 *The magnification factor of the d-cube $\gamma(G) = 1$.*

Proof:

1. $\gamma(G) \leq 1$.
   Let S_1 be a vertex set of "half cube" (e.g. all vertices with state number starting with 0). Then clearly, every vertex in S_1 will have exactly one edge incident to it leaving S_1. Therefore, $|\delta(S_1)| = |S_1| = \frac{|V|}{2}$, and

$$\gamma(G) = \min_{0<|S|\leq\frac{|V|}{2}} \frac{|\delta(S)|}{|S|} \leq \frac{|\delta(S_1)|}{|S_1|} = 1.$$

2. $\gamma(G) \geq 1$.
   For all x_1, x_2, define a random path between x_1 and x_2 selected uniformly among all shortest paths between x_1 and x_2. For example, for

$$\begin{array}{lcccccccc} x_1 & = & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ x_2 & = & 1 & 0 & 0 & 1 & 1 & 1 & 1 \end{array}$$

   we first look at the bits in which they differ (3rd, 5th and 7th). The order in which you change these bits determines a (or all) shortest path in the d-cube.

   Given $e \in E$, by symmetry,

$$E[\#\text{ of paths through e}] = \frac{1}{|E|} \cdot T,$$

   where T is the total length of all shortest paths. We can compute this by first choosing x_1 (2^d choices), sum up all the paths from x_1(from length 1 to d). For any path p, we count it twice: $x_1 \to x_2$ and $x_2 \to x_1$. Thus,

$$T = \frac{1}{2} \cdot 2^d \sum_{k=0}^{d} \binom{d}{k} k.$$

   This can also be written as $T = \frac{1}{2} \cdot 2^d \sum_{k=0}^{d} \binom{d}{k}(d-k)$. Taking the average of these two expressions, we deduce that

$$T = \frac{1}{2} \cdot d2^d \sum_{k=0}^{d} \binom{d}{k} = d2^{2d-2}.$$

   On the other hand, $|E| = \frac{1}{2}2^d \cdot d$. Hence,

$$E[\#\text{ of paths through } e] = \frac{T}{|E|} = \frac{d2^{2d-2}}{d2^{d-1}} = 2^{d-1}.$$

   Consider any set S. By linearity of expectations,

$$E[\#\text{ of paths intersecting } \delta(S)] \leq \sum_{e\in\delta(S)} E[\#\text{ of paths through } e] = 2^{d-1}|\delta(S)|.$$

However, the number of paths crossing $\delta(S)$ should be at least $|S| \cdot |\overline{S}|$. This implies that

$$|S| \cdot |\overline{S}| \le E[\# \text{ of paths through } \delta(S)] = 2^{d-1} \cdot |\delta(S)|.$$

Since $|S| \le \frac{|V|}{2}$, $|\overline{S}| \ge \frac{|V|}{2} = 2^{d-1}$. Therefore, for any set S,

$$\frac{|\delta(S)|}{|S|} \ge \frac{|\overline{S}|}{2^{d-1}} \ge 1.$$

So, $\gamma(G) \ge 1$.

□

This gives us the conductance of the corresponding MC: $\Phi = p \cdot \gamma(G) = \frac{1}{2d}$. Then $\lambda_2 \le 1 - \frac{\Phi^2}{2} = 1 - \frac{1}{8d^2}$. The steady-state distribution is $\pi_j = \frac{1}{2^d}$ for all j. Thus, the relative error after t steps is

$$\Delta = \max_{i,j} \frac{p_{ij}^t - \pi_j}{\pi_j} \le \frac{\lambda_2^t}{\min_j \pi_j} \le 2^d \cdot \left(1 - \frac{1}{8d^2}\right)^t.$$

If we want $\Delta \le \epsilon$, we shall choose t such that:

$$\begin{aligned} t &\ge \frac{d \ln 2 - \ln \epsilon}{-\ln(1 - \frac{1}{8d^2})} \\ &\ge 8d^2 \cdot (d \ln 2 - \ln \epsilon). \end{aligned}$$

In this case, although the MC has an exponential number of states (2^d), we only need $O(d^3)$ steps to generate an almost uniform state (with ϵ say constant or even as small as $e^{-O(d)}$).

In general, let M be an ergodic, time reversible Markov chain with $e^{q(n)}$ states, where $q(n)$ is a polynomial in n (n represents the size of the input). If its conductance $\Phi \ge \frac{1}{p(n)}$, where $p(n)$ is a polynomial in n, we will say that it has the *rapidly mixing property.* The relative error after t steps is

$$\Delta_t \le \frac{\lambda_2^t}{\min_j \pi_j} \le \frac{(1 - \frac{\Phi^2}{2})^t}{e^{q(n)}} \le \epsilon,$$

To get $\Delta_t \le \epsilon$, we only need to take $t = 2p^2(n)\left(q(n) + \ln \frac{1}{\epsilon}\right)$ steps, a polynomial number in n and $\ln \frac{1}{\epsilon}$.

Thus a rapidly-mixing MC with uniform stationary distribution with state space M can be used as an ϵ-sampling scheme on M:

Definition 5 *A* fully polynomial ϵ-sampling scheme *(also called an* ϵ-generator*) for a set M is an algorithm that runs in time poly(size of input, $\ln \frac{1}{\epsilon}$), and outputs an element $x \in M$ with probability $\pi(x)$ such that*

$$\max_{x \in M} \left| \pi(x) - \frac{1}{|M|} \right| \le \frac{\epsilon}{|M|}.$$

(M is typically given implicitly by a relation (i.e. on input x, M is the set of strings y satisfying some relation $< x, y >$).)

8 Approximation by sampling

We now sketch how an ϵ-sampling scheme can be used to develop a randomized approximation algorithm for the counting problem we discussed before. To evaluate $|M|$, first we immerse M into a larger set $V \supseteq M$. Then we sample from V, and approximate $\frac{|M|}{|V|}$ by

$$\frac{\text{size of } (M \cap \text{ sample})}{\text{size of sample}}.$$

This scheme works well if $|M|$ is polynomially comparable to $|V|$. But if $|M| \ll |V|$ (i.e. $|M|$ exponentially smaller than $|V|$), this scheme will have trouble, since in order to obtain a small relative error in the approximation, the number of samples will need to be so large (i.e. exponential) as to make this approach infeasible. (See our previous discussion for the problem of counting individuals, and our study of Chernoff bounds.)

Example: Suppose we wish to approximate π. If we take a square with side-length 2 and we inscribe within it a circle of radius 1, then the ratio of the area of the circle to the area of the square is $\pi/4$. Thus the probability that a uniformly generated point in the square belongs to the circle is precisely $\pi/4$.

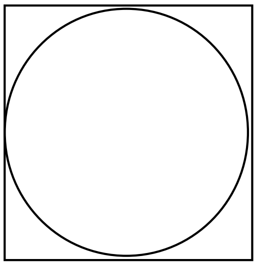

Figure 8: How (not) to calculate π.

By generating points within the square at random according to a uniform distribution, we can approximate π as simply 4 times the fraction of points that lie within the circle. The accuracy of such an approximation depends on how closely we can approximate the uniform distribution and on the number of samples. However, we will run into trouble if we want to estimate $vol(B_n)$ by using the same method, where $B_n = \{x \in R^n : ||x|| \leq 1\}$, since the volume is exponentially smaller than the volume of the corresponding cube (2^n). Nevertheless, a very nice application of rapidly mixing markov chains is precisely in the computation of the volume of a convex body or region [6]. To avoid the problem just mentioned, what is done is to immerse the body whose volume V_0 needs to be computed in a sequence of bodies of volumes V_1,

$V_2, \ldots$, such that V_i/V_{i+1} is is polynomially bounded. Then one can evaluate V_0 by the formula:

$$\frac{V_0}{V_n} = \frac{V_0}{V_1} \cdot \frac{V_1}{V_2} \cdot \frac{V_2}{V_3} \cdots \frac{V_{n-1}}{V_n}.$$

We now show how this technique can be used to develop a fully polynomial randomized approximation scheme for computing the permanent of a class of 0-1 matrices.

9 Approximating the permanent

Recall that for an $n \times n$ 0-1 matrix A, the *permanent* of A, $\mathrm{perm}(A)$, is the number of perfect matchings in the bipartite graph G whose incidence matrix is A. It is known that computing $\mathrm{perm}(A)$ is #P-complete.

In order to develop an approximation scheme for the permanent, we use the technique of approximation by sampling. As a naive adoption of this technique, we can generate edge sets at random and count the fraction that are perfect matchings. Unfortunately, this scheme may resemble searching for a needle in a haystack. If the fraction of edge sets that are perfect matchings is very small, then in order to obtain a small relative error in the approximation, the relative error in the sampling may need to be so small and the number of samples may need to be so large as to make this approach infeasible.

Instead of trying to directly approximate the fraction of edge sets that are perfect matchings, we can try to approximate a different ratio from which the permanent can be computed. Specifically, for $k = 1, 2, \ldots, n$, let $\mathcal{M}_k$ denote the set of matchings with size k, and let $m_k = |\mathcal{M}_k|$ denote the number of matchings with size k. The permanent of A is then given by $\mathrm{perm}(A) = m_n$, and we can express $\mathrm{perm}(A)$ as the product of ratios:

$$\mathrm{perm}(A) = \frac{m_n}{m_{n-1}} \frac{m_{n-1}}{m_{n-2}} \cdots \frac{m_2}{m_1} m_1 \tag{7}$$

($m_1 = |E|$). Thus, we can approximate the permanent of A by approximating the ratios m_k/m_{k-1} for $k = 2, 3, \ldots, n$. We write m_k/m_{k-1} as

$$\frac{m_k}{m_{k-1}} = \frac{u_k}{m_{k-1}} - 1 \tag{8}$$

where $u_k = |\mathcal{U}_k|$ and $\mathcal{U}_k = \mathcal{M}_k \cup \mathcal{M}_{k-1}$ (see Figure 9), and then we use an ϵ-sampling scheme for $\mathcal{U}_k$ to approximate the fraction m_{k-1}/u_k. To summarize our approach, for each $k = 2, 3, \ldots, n$, we take random samples over a uniform distribution on the set of matchings of size k and $k-1$, and we count the fraction that are matchings of size $k-1$; this gives us m_{k-1}/u_k, and we use Equation 8 to get m_k/m_{k-1}. Equation 7 then gets us $\mathrm{perm}(A)$.

The following two claims establish the connection between ϵ-sampling of $\mathcal{U}_k$ and approximation of the permanent of A.

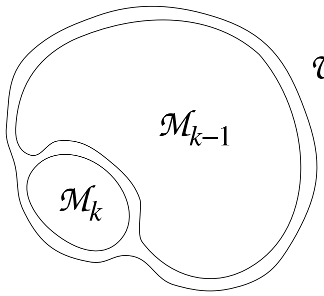

Figure 9: Each matching in $\mathcal{U}_k$ has size either k or $k-1$.

Claim 18 (Broder) *If there is a fully polynomial ϵ-sampling scheme for $\mathcal{U}_k = \mathcal{M}_k \cup \mathcal{M}_{k-1}$, then there exists a randomized approximation scheme for m_k/m_{k-1} that runs in time that is polynomial in the values $1/\epsilon$, n, and u_k/m_k.* ■

Claim 19 *If for each $k = 2, 3, \ldots, n$, there is a fully polynomial ϵ-sampling scheme for $\mathcal{U}_k$, then there exists a randomized approximation scheme for the permanent of A that runs in time that is polynomial in the values $1/\epsilon$, n, and $\max_k(u_k/m_k)$.* ■

For those graphs with $\max_k(u_k/m_k)$ bounded by a polynomial in n, Claim 19 gives us a fully polynomial randomized approximation scheme for the permanent — provided, of course, that we can produce an ϵ-sampling scheme for $\mathcal{U}_k$. In fact, it turns out that

$$\max_k \left\{ \frac{u_k}{m_k} \right\} = \frac{u_n}{m_n}.$$

This is because $\{m_k\}$ is log-concave (i.e. $m_k m_{k+2} \leq m_{k+1}^2$). Thus, if we can develop an ϵ-sampling scheme for the matchings $\mathcal{U}_k$ for each $k = 2, 3, \ldots, n$, then for the class of graphs with u_n/m_n bounded by a polynomial in n, we have an fpras for the permanent. After developing an ϵ-sampling scheme, we will look at such a class of graphs.

An ϵ-sampling scheme for matchings

We now turn our attention to developing an ϵ-sampling scheme for $\mathcal{U}_k = \mathcal{M}_k \cup \mathcal{M}_{k-1}$, and it should come as no surprise that we will use the technique of rapidly mixing Markov chains.

We now define a Markov chain whose states are matchings in $\mathcal{U}_k$. Consider any pair M_i, M_j of states (matchings) and create a transition between them according to the following rules:

- If M_i and M_j differ by the addition or removal of a single edge, that is, $M_i \triangle M_j = \{e\}$ for some edge e, then there is a transition from M_i to M_j and a transition from M_j to M_i. Both transitions have probability $p_{ij} = p_{ji} = 1/2m$ where m denotes the number of edges in the graph. $\big(M_i \triangle M_j = (M_i - M_j) \cup (M_j - M_i)\big)$

- If M_i and M_j are both matchings of size $k-1$ and they differ by removing one edge and adding another edge that has an endpoint in common with the removed edge, that is, $M_i, M_j \in \mathcal{M}_{k-1}$ and $M_i \triangle M_j = \{(u,v),(v,w)\}$ for some pair of edges (u,v) and (v,w), then there is a transition from M_i to M_j and a transition from M_j to M_i. Both transitions have probability $p_{ij} = p_{ji} = 1/2m$.

To complete the Markov chain, for each state M_i, we add a loop transition from M_i to itself with probability p_{ii} set to ensure that $\sum_j p_{ij} = 1$.

It is easy to see that this Markov chain is ergodic since the self-loops imply aperiodicity, and irreducibility can be seen from the construction. Irreducibility implicitly assumes the existence of some matching of size k; otherwise the chain might not be irreducible. The proof of irreducibility indeed stems on the fact that any matching of size k can reach any matching of size $k-1$, implying that one can reach any matching from any other matching provided a matching of size k exists. This Markov chain is time reversible and has the desired uniform steady state probability distribution, $\pi_i = 1/u_k$, since it is clearly symmetric. Furthermore, $p_{ii} \geq 1/2$ for each state M_i, and therefore, $\lambda_{\max} = \lambda_2$ which means that we can bound the relative error after t steps by:

$$\Delta_t \leq u_k \left(1 - \frac{\Phi^2}{2}\right)^t.$$

Finally, this Markov chain also has the property that $p_{ij} = p = 1/2m$ for every transition with $i \neq j$, and this property allows us to compute Φ by:

$$\Phi = \frac{1}{2m}\gamma(H)$$

where we recall that $\gamma(H)$ is the magnification of the underlying graph H (not the graph G on which we are sampling matchings).

If we could now lower bound $\gamma(H)$ by $\gamma(H) \geq 1/p(n)$ where $p(n)$ is a polynomial in n, then since $m \leq n^2$, we would have $\Phi \geq 1/p'(n)$ ($p'(n)$ is also a polynomial in n), and so we would have a fully polynomial ϵ-sampling scheme for $\mathcal{U}_k$. We cannot actually show such a lower bound, but the following theorem gives us a lower bound of $\gamma(H) \geq m_k/cu_k$ (c is a constant), and this, by Claim 19, is sufficient to give us a randomized approximation scheme that is polynomial in $1/\epsilon$, n, and u_n/m_n.

Theorem 20 (Dagum, Luby, Mihail and Vazirani [5]) *There exists a constant c such that*

$$\gamma(H) \geq \frac{1}{c}\frac{m_k}{u_k}.$$

Corollary 21 *There exists a fully polynomial ϵ-sampling scheme for $\mathcal{U}_k$ provided that $\frac{u_k}{m_k} = O(n^{c'})$ for some c'.*

A preliminary lemma is required.

Lemma 22 *Let M_1, M_2 be matchings in a bipartite graph. Then $M_1 \triangle M_2$ is a union of vertex disjoint paths and cycles which alternate between the edges of M_1 and M_2.*

Sketch of the proof of main result: For each pair of states M_1, M_2 with $M_1 \in \mathcal{M}_{k-1}$ and $M_2 \in \mathcal{M}_k$, we pick a random path from M_1 to M_2 in H as follows. By lemma 22 we can write the symmetric difference of the matchings M_1 and M_2 as

$$M_1 \triangle M_2 = C \cup D \cup E$$

where each element of C denotes a cycle or path in G with the same number of edges from M_1 as from M_2, each element of D denotes a path in G with one more edge from M_2 than from M_1, and each element of E denotes a path in G with one more edge from M_1 than from M_2. Notice that there must be exactly one more element in D than in E. We order the paths in each set at random so $[C_1, C_2, \ldots, C_r]$ is a random permutation of the r paths (or cycles) in C, $[D_1, D_2, \ldots, D_q]$ is a random permutation of the q paths in D, and $[E_1, E_2, \ldots, E_{q-1}]$ is a random permutation of the $q-1$ paths in E. A path from M_1 to M_2 in H is then given by:

$$\begin{tikzcd}
M_1 \arrow[dr, "D_1"] & \\
& M' = M_1 \triangle D_1 \arrow[dl, "E_1"] \\
M'' = M' \triangle E_1 & \\
\vdots \arrow[dr, "D_q"] & \\
& M''' \arrow[d, "C_1"] \\
& \vdots \arrow[d, "C_r"] \\
& M_2
\end{tikzcd}$$

Of course, $M_1 \to (M_1 \triangle D_1)$ may not actually be a transition of the chain, but $M_1 \to (M_1 \triangle D_1)$ does define a path of transitions if we use the edges of D_1 two at a time.

The crucial part of this proof is showing that there exists a constant c such that for any edge e of H, the expected number of paths that go through e is upper bounded by

$$\mathrm{E}\,[\text{number of paths through } e] \leq c m_{k-1}.$$

We will not do this part of the proof. Now if we consider any set $S \subseteq V$ of vertices from H, by linearity of expectation, the expected number of paths that cross from S over to $\overline{S}$ is upper bounded by

$$\mathrm{E}\left[\text{number of paths crossing } S\right] \leq c m_{k-1} \left|\delta(S)\right|$$

where we recall that $\delta(S)$ denotes the coboundary of S. Therefore, there exists some choice for the paths such that not more than $c m_{k-1} \left|\delta(S)\right|$ of them cross the boundary of S.

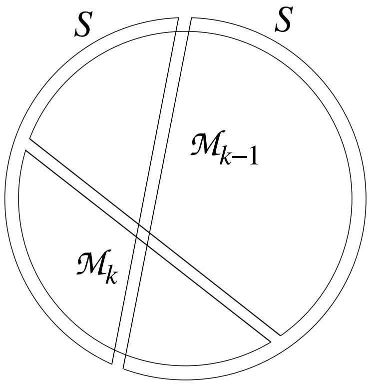

Figure 10: Partitioning $\mathcal{U}_k$.

Each vertex of S is either a matching of $\mathcal{M}_k$ or a matching of $\mathcal{M}_{k-1}$ and likewise for $\overline{S}$, so we can partition $\mathcal{U}_k$ as shown in Figure 10. We assume, without loss of generality, that

$$\frac{|S \cap \mathcal{M}_k|}{|S|} \geq \frac{m_k}{u_k}, \tag{9}$$

and therefore,

$$\frac{\left|\overline{S} \cap \mathcal{M}_{k-1}\right|}{\left|\overline{S}\right|} \geq \frac{m_{k-1}}{u_k} \tag{10}$$

(otherwise, we exchange S and $\overline{S}$). The number of paths that cross S must be at least $|S \cap \mathcal{M}_k||\overline{S} \cap \mathcal{M}_{k-1}|$ since for any $M_1 \in \overline{S} \cap \mathcal{M}_{k-1}$, $M_2 \in S \cap \mathcal{M}_k$ there must be a path from M_1 to M_2 which crosses from S to $\overline{S}$. By multiplying together Inequalities 9 and 10, we have

$$|S \cap \mathcal{M}_k| \left|\overline{S} \cap \mathcal{M}_{k-1}\right| \geq \frac{m_k m_{k-1} |S| \left|\overline{S}\right|}{u_k^2}.$$

But we have already seen that we can choose paths so that the number of paths crossing the boundary of S is not more than $cm_{k-1}|\delta(S)|$. Therefore, it must be the case that

$$cm_{k-1}|\delta(S)| \geq \frac{m_k m_{k-1}|S|\left|\overline{S}\right|}{u_k^2}.$$

Notice that this statement is unchanged if we replace S by $\bar{S}$. So, without loss of generality $|S| \leq \frac{u_k}{2}$ which implies that $\left|\overline{S}\right| \geq \frac{u_k}{2}$. Hence

$$\frac{m_k}{u_k^2}|S|\left|\overline{S}\right| \geq \frac{m_k}{u_k}\frac{|S|}{2}$$

which implies that

$$\frac{|\delta(S)|}{|S|} \geq \frac{1}{2c}\frac{m_k}{u_k}$$

$$\gamma(H) \geq \frac{1}{2c}\frac{m_k}{u_k}.$$

■

A class of graphs with u_n/m_n polynomially bounded

We finish this discussion by considering a class of graphs for which u_n/m_n is bounded by a polynomial in n. Specifically, we consider the class of *dense* bipartite graphs. A dense bipartite graph is a bipartite graph in which every vertex has degree at least $n/2$ (recall that n is number of vertices on each side of the bipartition).

We now show that for dense bipartite graphs, $m_{n-1}/m_n \leq n^2$. Since $u_n/m_n = 1+m_{n-1}/m_n$, this bound gives us the desired result. Consider a matching $M \in \mathcal{M}_{n-1}$ with edges $\{(u_1, v_1), (u_2, v_2), \ldots, (u_{n-1}, v_{n-1})\}$ so that u_n and v_n are the two exposed vertices. Since both u_n and v_n have degree at least $n/2$, there are the following two possibilities.

- (u_n, v_n) is an edge which implies that $M' := M \cup \{(u_n, v_n)\} \in \mathcal{M}_n$.
- There exists an i for $1 \leq i \leq n-1$ such that both (u_n, v_i) and (u_i, v_n) are edges (this follows from the pigeonhole principle). In this case $M' := M - \{(u_i, v_i)\} \cup \{(u_n, v_i), (u_i, v_n)\} \in \mathcal{M}_n$.

Thus we can define a function $f : \mathcal{M}_{n-1} \to \mathcal{M}_n$ by letting $f(M) = M'$.

Now consider a matching $M' \in \mathcal{M}_n$, and let $f^{-1}(M')$ denote the set of matchings $M \in \mathcal{M}_{n-1}$ such that $f(M) = M'$. For each $M \in \mathcal{M}_{n-1}$ such that $f(M) = M'$, M can be obtained from M' in one of two different ways.

- Some pair of edges $(u_i, v_i), (u_j, v_j)$ are removed from M' and replaced with a single edge that must be either (u_i, v_j) or (u_j, v_i). We can choose the pair of edges to remove in $\binom{n}{2}$ ways and we can choose the replacement edge in at most 2 ways.

- An edge (u_i, v_i) is removed from M'. This edge can be chosen in n ways.

Thus, there are at most

$$2\binom{n}{2} + n = n(n-1) + n = n^2$$

matchings in $\mathcal{M}_{n-1}$ that could possibly map to M'. This means that $|f^{-1}(M')| \leq n^2$ for every matching $M' \in \mathcal{M}_n$, and therefore, $m_{n-1}/m_n \leq n^2$.

Thus we have shown that for dense bipartite graphs, there exists a fully polynomial randomized approximation scheme for the permanent. This result is tight in the sense that graphs can be constructed such that $\frac{m_{n-1}}{m_n}$ is exponential and whose vertices have degree $\geq \frac{n}{2} - \epsilon$. There is also a theorem of Broder which says that

Theorem 23 *Counting perfect matchings on dense bipartite graphs is #P-complete.*

Other applications of Markov Chains

There are other uses for Markov Chains in the design of algorithms. Of these the most interesting is for computing the volume of convex bodies. It can be shown that a fully polynomial randomized approximation scheme may be constructed for this problem [6]. This is interesting because it can also be shown that one cannot even approximate the volume to within an exponential factor of n^{cn} for $c < 0.5$ in polynomial time, where n is the dimension [3].

References

[1] N. Alon. Eigenvalues and expanders. *Combinatorica*, 6:83–96, 1986.

[2] N. Alon and V. Milman. λ_1, isoperimetric inequalities for graphs and superconcentrators. *Journal of Combinatorial Theory B*, 38:73–88, 1985.

[3] I. Bárány and Z. Füredi. Computing the volume is difficult. In *Proceedings of the 18th Annual ACM Symposium on Theory of Computing*, pages 442–447, 1986.

[4] J. Cheeger. A lower bound for the smallest value of the Laplacian. In R. Gunning, editor, *Problems in analysis*, pages 195–199. Princeton University Press, 1970.

[5] P. Dagum, M. Mihail, M. Luby, and U. Vazirani. Polytopes, permanents and graphs with large factors. In *Proceedings of the 29th Annual Symposium on Foundations of Computer Science*, pages 412–422, 1988.

[6] M. Dyer, A. Frieze, and R. Kannan. A random polynomial algorithm for approximating the volume of convex bodies. *Journal of the ACM*, pages 1–17, 1991.

[7] R. Karp. An introduction to randomized algorithms. *Discrete Applied Mathematics*, 34:165–201, 1991.

[8] R. M. Karp and M. O. Rabin. Efficient randomized pattern-matching algorithms. *IBM Journal of Research and Development*, 31:249–260, 1987.

[9] R. Motwani and P. Raghavan. *Randomized Algorithms*. 1994.

[10] K. Mulmuley, U. Vazirani, and V. Vazirani. Matching is as easy as matrix inversion. *Combinatorica*, 7(1):105–113, 1987.

[11] A. Sinclair and M. Jerrum. Approximate counting, uniform generation and rapidly mixing markov chains. *Information and Computation*, 82:93–133, 1989.