CS 683: Advanced Algorithms	**Approximate counting**
Tue, 03/27/2001	**Scribe: Alex Slivkins**

In the next two lectures we'll use a randomized algorithm for approximated counting. The "crux" of the algorithm we are going to use is that it selects approximately uniformly random elements from a certain set of exponential size. The challenge is to do it in polynomial time. We'll accomplish this via random walks, using the machinery we developed in the previous two lectures.

This lecture we'll introduce a certain "hard" counting problem, give a polynomial-time approximating algorithm and take care of all details except approximately uar selection.

1 Permanents and perfect matchings

1.1 Permanent of a matrix

Let A be a $n \times n$ matrix. Let S_n be a set of permutations on $\{1 \quad , \dots \quad ,n\}$. Recall that the determinant of A is defined as

$$\textsc{det}(A) = \sum_{\pi \in S_n} \textsc{sign}(\pi) \prod_{j=1}^{n} A[j, \pi(j)]$$

What if we remove $\textsc{sign}(A)$?

Definition 1 *The permanent* $\textsc{perm}$*(A) is*

$$\textsc{perm}(A) = \sum_{\pi \in S_n} \prod_{j=1}^{n} A[j, \pi(j)]$$

Although $\textsc{det}(A)$ is polynomial, $\textsc{perm}(A)$ turns out NP-hard. How could we prove it? Usually NP-hard problems about matrices stay NP-hard even if restricted to 0-1 matrices. So assume A is a 0-1 matrix. Then

$$\begin{aligned} \textsc{perm}(A) &= |\{\pi \in S_n \mid \prod_{j=1}^{n} A[j, \pi(j)] = 1\}| \\ &= |\{\pi \in S_n \mid A[j, \pi(j)] = 1, \ \forall j\}| \end{aligned}$$

Let X be the set of rows, Y the set of columns of A. Construct a bipartite graph $G = (X, Y, E)$, where for $i \in X$, $j \in Y$ edge $(i, j) \in E$ iff $A[i, j] = 1$. Then permutations $\pi \in S_n$ s.t. $A[j, \pi(j)] = 1, \ \forall j$ are in 1-1 correspondence with perfect matchings in G. Therefore,

$$\textsc{perm}(A) = \#\textsc{pm}(G)$$

where $\#\text{PM}(G)$ is the number of perfect matchings in G. Moreover, computing $\text{PERM}(A)$ is really no easier than counting perfect matchings, because given a bipartite graph $G = (X, Y, E)$ s.t. $|X| = |Y|$ one can easily reconstruct A.

To prove something is NP-hard, we need to turn it into a decision problem first. The natural question, $[\text{PERM}(A) \stackrel{?}{=} 0]$, does not work out, since there is a polynomial algorithm to determine whether a bipartite graph has a perfect matching.

Can try $[\#\text{PM} \stackrel{?}{<} N]$. This problem must be NP-hard if $\text{PERM}(A)$ is. Why? Using a subroutine that solves $[\#\text{PM} \stackrel{?}{<} N]$, we can determine $\#\text{PM}$ by binary search. Since $\#\text{PM}$ is at most exponential, binary search would run in a polynomial time.

Alternatively, we can introduce a more general notion of intractability.

1.2 Complexity class #P

Definition 2 *Let #P be the class of problems of the following form: given a non-deterministic polynomial-time bounded verifying machine N, return the number of accepting computations of N.*

Definition 3 *A problem #P-hard if all problems in #P can be reduced to it via polynomial-time reductions. A problem is #P-complete if it is #P and #P-hard.*

Note that any NP problem trivially reduces to #P: count the number x of accepting computations and accept iff $x > 0$.

Theorem 4 *#SAT is #P-complete.*

Proof: The proof of Cook's theorem gives a way to encode, in polynomial time, a given polynomial-time non-deterministic Turing machine N as a boolean formula $\phi(N)$, so that there is a 1-1 correspondence between accepting computations of N and satisfying assignments of $\phi(N)$. $\square$

#SAT is the "main" #P-complete problem, same way as SAT is for NP.

Theorem 5 *(w/o proof)* $\#\text{PM}$ *is #P-complete.*

This theorem is surprising because $[\#\text{SAT} \stackrel{?}{=} 0]$ is just co-SAT, thus co-NP-complete, whereas a polynomial algorithm is known for $[\#\text{PM} \stackrel{?}{=} 0]$.

1.3 Goal

We have proved that PERM is #P-hard even for 0-1 matrices. Now we'll construct an approximate algorithm. We'll restrict PERM to *dense*[1] 0-1 matrices. In this case $\text{PERM}(A) =$

[1] 0-1 matrix is dense if the sum of any row or column is at least $n/2$.

#PM(G) for a *dense*[2] bipartite graph $G = (X, Y, E)$ s.t. $|X| = |Y|$. As we proceed it will become clear why we need A or G to be dense.

In what follows we'll 'forget' about PERM's and think only about #PM. We want to approximate #PM to within a factor of $1 \pm \epsilon$ by a randomized algorithm with running time polynomial in the size of a graph and $1/\epsilon$. Let's state it more formally:

Goal: Randomized algorithm that

- takes a number $\epsilon > 0$ and a dense bipartite graph $G = (X, Y, E)$ s.t. $|X| = |Y|$,
- runs in time polynomial in $1/\epsilon$ and $n = |X| = |Y|$,
- returns a number x s.t. with high probability

$$1 - \epsilon \leq \frac{x}{\#\text{PM}(G)} \leq 1 + \epsilon \tag{1}$$

This problem was solved in [1]. The case of general 0-1 matrices was solved some 10 years later, in [2], but we won't look into it here.

At this point we are looking for *any* polynomial algorithm, not necessarily for a fast one. The algorithm we'll present is pretty bad in terms of the actual running time, $O(n^{15})$ or something like that, but still polynomial.

2 The algorithm

2.1 Approach

Let $G = (X, Y, E)$ be a dense bipartite graph s.t. $|X| = |Y|$. Let $n = |X| = |Y|$. Let $\mathcal{M}_k$ be the set of all size k matchings in G, $m_k = |\mathcal{M}_k|$. The goal is to compute $m_n = \#\text{PM}$. Our method is to express m_n as a telescopic product of ratios:

$$m_n = m_1 \times \frac{m_2}{m_1} \times \frac{m_3}{m_2} \cdots \times \frac{m_n}{m_{n-1}}$$

Since $m_1 = |E|$, it remains to estimate the ratios $r_k = m_k/m_{k-1}$.

Consider the set $\mathcal{M}_{k-1} \cup \mathcal{M}_k$ and repeatedly generate elements uniformly at random (uar) from this set. Then r_k is approximated by

$$\frac{\#\text{ generated elements in } \mathcal{M}_k}{\#\text{ generated elements in } \mathcal{M}_{k-1}}$$

$\mathcal{M}_k$'s might be exponentially large, so we have three problems:

(1) how to generate elements uar?

(2) if r_k is big, we might need exponentially many samples to get the desired approximation.

(3) errors might 'telescope': small errors in the beginning might become large eventually.

[2] A graph is dense if degree of each node is at least $n/2$.

As Jon puts it, (1) is 'really the crux of it'. This lecture we'll show we only need the selection to be *approximately* uar; we'll show exactly what "quality" of approximation suffices. Next lecture we'll use random walks to get the approximation of desired quality.

We'll take care of (2) and (3) later this lecture; we'll deal with (2) by proving that it cannot happen if G is dense. This is where this condition becomes handy. (3) is the easiest, so we'll do it first.

2.2 Telescoping errors

Suppose there exists[3] a B s.t.

$$1/B \leq r_k \leq B \tag{2}$$

Say our estimate of r_k is r'_k s.t. for some $\gamma \in (0,1)$ with some high probability $1-p$

$$1-\gamma \leq \frac{r'_k}{r_k} \leq 1+\gamma \tag{3}$$

Our estimate of m_n is $m'_n = |E| r'_1 \cdots r'_n$. With probability at least $1-np$ Eq. (3) holds for all k. In this event,

$$1-\epsilon \overset{?}{\leq} (1-\gamma)^n \leq \frac{m'_n}{m_n} \leq (1+\gamma)^n \overset{?}{\leq} 1+\epsilon$$

Since

$$(1-\gamma)^n = [(1-\gamma)^{1/\gamma}]^{\gamma n} > e^{-\gamma n} > 4^{-\gamma n}$$

to get $(1-\gamma)^n \geq 1+\epsilon$ it suffices to have

$$\gamma \geq \frac{-\log_4(1-\epsilon)}{n} = \Theta(\epsilon/n)$$

Similarly, $\gamma = \Theta(\epsilon/n)$ suffices for $(1+\gamma)^n \leq 1+\epsilon$, too. Therefore, taking $\gamma = \Theta(\epsilon/n)$ suffices to make sure that with probability $1-np$ Eq. (1) holds.

Note that we also need the probability $1-np$ to be large.

2.3 How many samples do we need?

Given k and γ, we'll to choose polynomially many samples so that $p \leq 1/n^2$ and Eq. (3) holds.

Each sample is a 0-1 random variable X_i s.t.

$$X_i = \begin{cases} 1, & \text{is the sample is in } \mathcal{M}_k \\ 0, & \text{is the sample is in } \mathcal{M}_{k-1} \end{cases}$$

[3] Later, in Lemma 6, we'll show that $B = n^2$ works.

Our estimate of r_k is $r'_k = \frac{X}{N-X}$, where $X = \sum X_i$ and $N = \#\text{samples}$. Let

$$q_k = \frac{r_k}{1 + r_k}$$

Suppose for some $\delta \in (0,1)$ the mean $\mu = E(X) = \sum E(X_i)$ is s.t.

$$1 - \delta \leq \frac{\mu}{N q_k} \leq 1 + \delta \tag{4}$$

Let $\mathcal{E}$ be the event

$$\{1 - \delta \leq \frac{X}{\mu} \leq 1 + \delta\}$$

If δ is small enough, then under $\mathcal{E}$ Eq. (3) holds. By a long and ugly calculation (which we skip), $\delta = \frac{\gamma}{4B}$ suffices.

Let $1 - q$ be the probability of $\mathcal{E}$. It remains to choose N so that $q \leq 1/n^2$. By Chernoff bounds,

$$\Pr[X < (1-\delta)\mu] < e^{-\mu\delta^2/2} \overset{?}{\leq} 1/n^2$$

Solving this gives

$$\mu \geq \frac{4\log(n)}{\delta^2}$$

By Eq. (4), it suffices to have

$$N \geq \frac{4\log(n)}{q_k(1-\delta)\delta^2}$$

By Eq. (2), $q_k \geq \frac{1}{B(B+1)}$. Plug in $\delta = \frac{\gamma}{4B}$. So if B is polynomial in n, so can be N.

2.4 Remaining problems

- We still need to find a B that satisfies Eq. (2) and is polynomial in n.
- For any given $\delta \in (0,1)$ we need a polynomial-time algorithm to generate samples that satisfy Eq. (4) (next lecture).

2.5 Finding a B

Lemma 6 *If G is a dense bipartite graph then* $(\forall k)$ $1/n^2 \leq r_k \leq n^2$.

Proof: Recall $r_k = m_k/m_{k-1}$. We need to prove that $\mathcal{M}_k$ is not much bigger than $\mathcal{M}_{k-1}$ and vice versa. Suppose we need to prove that set A is not much bigger than set B, and there exists a multiple-valued map from A to B s.t.

- $\circ$ Each element of A is mapped to at least one element in B,
- $\circ$ $\forall b \in B$ at most x elements of A map to b.

Then $|A| \leq x|B|$. This is the approach we are going to use.

- Show $|\mathcal{M}_k| \leq n^2|\mathcal{M}_{k-1}|$.

  Map each $M \in \mathcal{M}_k$ to all $M' \in \mathcal{M}_{k-1}$ obtained by deleting an edge from M. At most m edges could have been deleted to make $M' \in \mathcal{M}_{k-1}$, so at most $m \leq n^2$ matchings in $\mathcal{M}_k$ can map to a single element of $\mathcal{M}_{k-1}$.

  This was an "easy" direction. In particular, it didn't require G to be dense.

- Show $|\mathcal{M}_{k-1}| \leq n^2|\mathcal{M}_k|$.

  Map each $M' \in \mathcal{M}_{k-1}$ to all $M \in \mathcal{M}_k$ obtained from M' by augmenting along an augmenting path of size ≤ 3. At least one such path exists by Lemma 7 (this is where we use the fact that G is dense!). Fix $M \in \mathcal{M}_k$. If $M' \in \mathcal{M}_{k-1}$ maps to a M then one of the following holds:

  - $\circ$ M' is M without an edge (k choices),
  - $\circ$ M' is M with two edges replaced by a "cross" edge (Fig. 1).

  There are $\frac{k(k-1)}{2}$ pairs of edges in M, times at most two ways to choose a "cross" edge, so at most $k + 2 \times \frac{k(k-1)}{2} = k^2 \leq n^2$ choices for M'. Therefore, at most n^2 different $M' \in \mathcal{M}_{k-1}$ map to a given $M \in \mathcal{M}_k$.

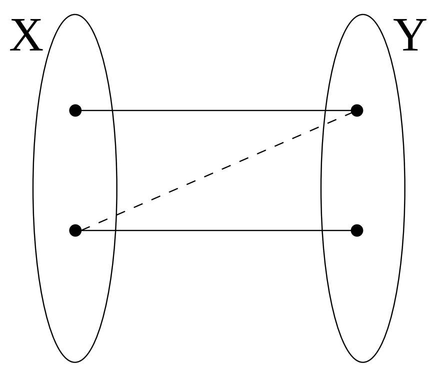


Figure 1: M is constructed from M' by augmenting along a path of length 3.

Lemma 7 *If G is dense, then every $M' \in \mathcal{M}_{k-1}$ has an augmenting path of length* 1 *(an edge) or* 3 *(Z-shape).*

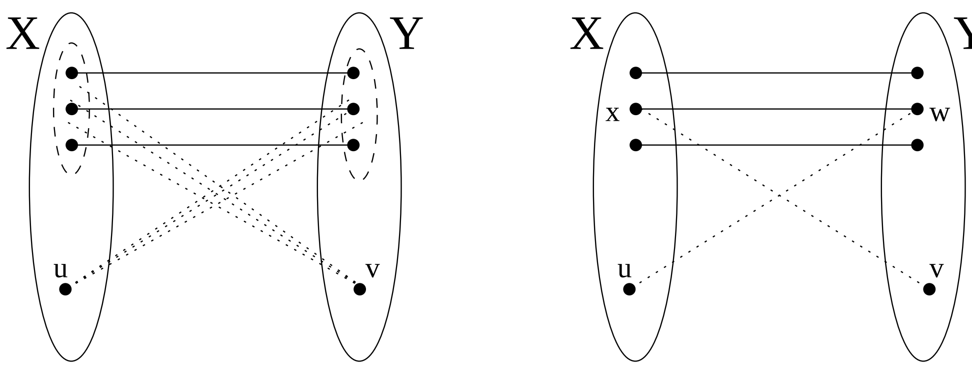
Proof: Fix $M' \in \mathcal{M}_{k-1}$. Since $\mathcal{M}_{k-1}$ is not a perfect matching, there are unmatched nodes $u \in X$, $v \in Y$. Suppose there is no augmenting path of length 1. Then any two unmatched nodes in M' are not adjacent in G. In particular, all neighbors of u and v are matched (Fig. 2a).

In other words, let $N(u)$, $N(v)$ be the sets of neighbors of u, v, resp. Let Z be the set of all matched nodes in M'. Then $N(u) \subset Z \cap Y$ and $N(v) \subset Z \cap X$.

Since

- $|Z \cap X| = |Z \cap Y| = k - 1 \leq n - 1$
- G is dense $\Rightarrow |N(u)| \geq n/2$ and $|N(u)| \geq n/2$

by Pigeonhole Principle there exist $w \in N(u)$, $x \in N(v)$ s.t. $(w, x) \in M'$ (Fig. 2b). Therefore, (u, w, x, v) is an augmenting path of length 3. $\square$



(a) (b)

(a) All neighbors of u and v are matched.
(b) There exist $w \in N(u)$, $x \in N(v)$ s.t. $(w, x) \in M'$.

Figure 2: For the proof of Lemma 7.

References

[1] Mark Jerrum and Alistair Sinclair. *Approximating the Permanent*. SIAM J. Computing 18(1989), pp. 1149-1178.

[2] Mark Jerrum, Alistair Sinclair, Eric Vigoda. *A polynomial-time approximation algorithm for the permanent of a matrix with non-negative entries*. ECCC Report TR00-079 (2000).