CS395T Psudorandomness and Combinatorial Constructions
Lecture 1, January 16, 2001
Lecturer: Prof. David Zuckerman, Scribe: Li-San Wang

1 Uses of Randomization

1. Monte Carlo simulations
2. Cryptography: generate private keys; randomness is provably necessary.
3. Distributed computing: breaking deadlocks; randomness is provably necessary.
4. Historically, the first randomized algorithm was primality testing. It is unknown whether a deterministic polynomial algorithm exists. Randomness is not provably necessary for algorithms; this issue will be explored later in this course.
5. The following example is the *Bipartite Matching* problem. This algorithm is randomized, and can be parallelized easily. The deterministic algorithm currently known cannot be parallelized. It is open whether there exists a deterministic algorithm that can be parallelized.

2 Example of Randomized Algorithm: Bipartite Matching

2.1 Definition of the Bipartite Matching problem

Input: A bipartite graph G, $V(G) = V_1 \cup V_2$, $|V_1| = |V_2| = n$, $V_1 \cap V_2 = \phi$.

Output: 1 if G has perfect matching, 0 otherwise. (A perfect matching is a set of edges in G such that each vertex in G is incident to exactly one edge in the matching.)

2.2 The randomized algorithm

The following is the randomized algorithm for the Bipartite Matching problem. Notation: $X \in_{\mathcal{U}} \mathbf{Y}$ means X is chosen from the set $\mathbf{Y}$ uniformly at random.

1. Associate an $n \times n$ matrix M_G with G as follows:
   (a) $(M_G)_{ij}{=}X_{ij}$ if the edge $(i,j) \in E(G)$,
   (b) $(M_G)_{ij}{=}0$ if the edge $(i,j) \notin E(G)$.
2. Notice that $\det(M_G) = P(X_{11}, \ldots, X_{nn})$ is a polynomial in n^2 variables. Verify if $P(X_{11}, \ldots, X_{nn})$ is a zero polynomial by the following steps:
   (a) Pick a prime $q > 2n$.
   (b) Pick $Y_{ij} \in_u \{0, 1, \ldots, q-1\}$. Compute $r = P(Y_{11}, \ldots, Y_{nn}) (\bmod\ q)$.
3. If r is a nonzero value, return 1, otherwise return 0.

Comments

1. $P(X_{11}, \ldots, X_{nn})$ is a polynomial consisting of $O(n!)$ monomials. It would be too costly to directly compute P.

2. There are polynomial time algorithms for computing the determinant of a matrix, where each entry is a value, not an indeterminate.

2.3 Correctness of the algorithm

The correctness is divided into two theorems. Theorem 1 establishes the validity of reducing bipartite matching to the determinant problem in the algorithm. Theorem 2 shows the correctness of the randomized argument in step 2 and 3 of the algorithm.

Theorem 1 $\det(M_G)$ *is not the zero polynomial iff G has a perfect matching.*

Proof:

$$\det(M_G) = \sum_{\sigma} \operatorname{sgn}(\sigma) \prod_{i=1}^{n} X_{i,\sigma(i)}$$

where the index σ runs over the $n!$ permutations of $\{1, \ldots, n\}$. Note that each term $\prod_{i=1}^{n} X_{i,\sigma(i)}$ corresponds to a possible perfect matching: $\{(i, \sigma(i)) : 1 \leq i \leq n\}$. So $\det(M_G)$ is a nonzero polynomial if and only if there is at least one term $\prod_{i=1}^{n} X_{i,\sigma(i)}$ that is not identically zero, which means there is a set of edges $\{(i, \sigma(i) : 1 \leq i \leq n, \sigma \text{ is a permutation}\}$. □

Corollary 1 *If $\det(M_G)$ is the zero polynomial, then $r = 0$ with probability 1.*

Before proving Theorem 2, we need the following lemma.

Lemma 1 (Schwartz-Zippel) *Let $P(X_1, \ldots, X_n)$ be a nonzero polynomial over any field $\mathcal{F}$ with total degree $\leq d$. Given any subset S of $\mathcal{F}$, $|S| = m$, Choose each $Z_i \in_u S$, then* $\Pr(P(Z_1, \ldots, Z_n) = 0) \leq \frac{d}{m}$.

Proof: The proof follows by induction on n, the number of indeterminates in p.

1. $n = 1$: $P(X_1) = 0$ has at most d roots, so $\Pr(P(Z_1) = 0) \leq \frac{d}{m}$.

2. Assume the claim holds for $n = k - 1$. Consider $n = k$. Let l be the highest degree of Z_k. We rewrite P as follows:

$$P(Z_1, \ldots, Z_k) = \sum_{j=1}^{l} P_j(Z_1, \ldots, Z_{k-1}) Z_k^j$$

Fix the first $k-1$ values $Z_1, \ldots, Z_{k-1}$. If $P_l(Z_1, \ldots, Z_{k-1}) \neq 0$ then $P(Z_1, \ldots, Z_k)$ is a nonzero polynomial of degree l. Therefore

$$\begin{aligned}
& \Pr(P(Z_1, \ldots, Z_k) = 0) \\
= \; & \Pr(P(Z_1, \ldots, Z_k) = 0, P_l(Z_1, \ldots, Z_{k-1})) + \Pr(P(Z_1, \ldots, Z_k) = 0, P_l(Z_1, \ldots, Z_{k-1}) \neq 0) \\
\leq \; & \Pr(P_l(Z_1, \ldots, Z_{k-1}) = 0) + \Pr(P(Z_1, \ldots, Z_k) = 0 | P_l(Z_1, \ldots, Z_{k-1}) \neq 0) \\
\leq \; & \frac{d-l}{m} + \frac{l}{m} = \frac{d}{m}
\end{aligned}$$

The second term in the third line is obtained from the fact given two events A and B, $\Pr(A,B) = \Pr(B|A)\Pr(A) \leq \Pr(B|A)$.

$\square$

Theorem 2 *If* $\det(M_G)$ *is a nonzero polynomial, the value* r *is nonzero with probability* $\geq \frac{1}{2}$.

Proof: Since $\mathbf{Z}_q = \{0, 1, \ldots, q-1\}$ is a field (with arithmetic modular q) when q is prime, we can use Lemma 1. If $\det(M_G)$ is not the zero polynomial, then $\Pr(r = 0) \leq \frac{n}{2n} = \frac{1}{2}$. $\square$

3 Randomization and Complexity Classes

The algorithm shows Bipartite Matching is in the complexity class **RP**, which stands for *randomized polynomial time*. Consider a set of languages L, for which there exists a deterministic Turing Machine $M_L(x,y)$. Here x is the actual input, and y is a string of independent random bits. We can define several complexity classes as follows.

Condition	Quantity to be bounded	**RP**	**P**	**NP**	**BPP**	**PP**
$x \in L$	$\Pr_y(M_L(x,y) = 1)$	$\geq \frac{1}{2}$	$= 1$	> 0	$\geq \frac{2}{3}$	$> \frac{1}{2}$
$x \notin L$	$\Pr_y(M_L(x,y) = 1)$	$= 0$	$= 0$	$= 0$	$\leq \frac{1}{3}$	$\leq \frac{1}{2}$

Comments:

1. To argue that the bounds for **NP** are correct, note that an input x is in L iff there exists a certificate that verifies this fact. An example of a certificate is a satisfying assignment for SAT. By choosing a potential certificate uniformly at random and using it to try to verify $x \in L$, we see that the verification occurs with positive probability iff $x \in L$.

2. To improve the error bound of the random classes, we can run the algorithm several times with independent choice of y, and take the majority.

3. The class **RP** has *1-sided error*, where the class **BPP** has *2-sided error*.

4. From the definition above we can see $\mathbf{P} \subseteq \mathbf{RP} \subseteq \mathbf{NP} \subseteq \mathbf{EXP}$, $\mathbf{P} \subseteq \mathbf{BPP} \subseteq \mathbf{EXP}$.

5. Similar concepts can be extended to numerical problems. Given a function A on input x, we look for an algorithm $M_A(x,y)$, where y is a string of random bits, such that $\Pr(|M_A(x,y) - A(x)| \leq \epsilon) \geq \rho$ for some value $\rho > \frac{1}{2}$. To improve the probability of error, we run the algorithm several times and take the *median*.

6. Open questions:
   (a) Show $\mathbf{RP} \neq \mathbf{EXP}$.
   (b) Show $\mathbf{BPP} \neq \mathbf{NEXP}$.