COMS 6998: Randomness in Computing — Sep 10, 2011

Basics of Probability Theory

Instructor: *Xi Chen* — Scribes: *Anthi Orfanou*

1 Introduction

The purpose of this course is to study probabilistic tools and their applications in Computer Science. Several topics that will be covered are given bellow:

- Verification/Testing (e.g. polynomial identity testing (PIT)).
- Using randomness to avoiding worst cases / dealing with smart adversaries (e.g. perfect hashing, probabilistic analysis, smoothed analysis of algorithms).
- Random walks/mixing time.
- Randomness presented in the problem (e.g. input drawn from a distribution).
- Probabilistic Method (e.g. 3SAT if you choose randomly truth values for the variables, then with probability > 0 at least $7/8$ of the clauses are satisfied, implying the existence of a truth assignment that satisfies at least $7/8$ of the clauses).

Definition 1. Probability Space.

1. *Sample space Ω: the set of all possible outcomes.*
2. *Probability function: $Pr : \Omega \to \mathbb{R}_+$ such that $\sum_{e\in\Omega} Pr(e) = 1$.*
3. *Event $E \subseteq \Omega$ such that $Pr[E] = \sum_{e\in E} Pr(e)$.*

2 Polynomial Identity Testing

Definition 2. *Polynomial Identity Testing (PIT). Given a polynomial $f \in \mathbb{R}[x]$ of degree d decide if $f \equiv 0$ or not. Alternatively we are given two polynomials $f, g \in \mathbb{R}[x]$ of degree d and we wish to decide if they are equal.*

In the above problem we assume that the polynomial is given as a product formula, e.g. $f(x) = (x+1)(x-2)\dots(x-6)+1$, otherwise it would be easy to decide it by simply checking the coefficients of the form $f(x) = a_d x^d + \cdots + a_1 x + a_0$. In the hardest scenario we are given black box access to the polynomial f, i.e. for every input x we ask, we obtain the answer $f(x)$. A trivial deterministic algorithm that solves the problem is to ask for $d+1$ values $f(1), f(2), \dots, f(d+1)$ that determine whether the polynomial of degree d is the zero polynomial or not. We are now ready to give an easy randomised

algorithm for the above problem.

Randomized PIT:

- Pick $x \in \{1, 2, \ldots, 2d\}$ uniformly at random.
- Evaluate $f(x)$.
- Output "zero" if $f(x) = 0$ or "non-zero" if $f(x) \neq 0$.

In the above scheme we have $\Omega = \{1, 2, \ldots, 2d\}$ and $\forall\ i \in \Omega\ Pr(i) = \frac{1}{2d}$. In the case that $f \equiv 0$ the algorithm is always correct. Otherwise, if $f \not\equiv 0$ and $f(x) = 0$ the algorithm fails as x is a root, found with probability $Pr[f(x) = 0] \leq \frac{d}{2d} = 1/2$ which we want to enhance. Thus the Randomized PIT algorithm always gives the correct answer with probability at least $1/2$.

2.1 String comparison

An application of the PIT algorithm is to decide if two strings are equal, by deciding equality between two polynomials. We assume that we have two entities that communicate, say Alice and Bob, respectively possessing the strings $a_0a_1 \ldots a_n \in \{0,1\}^{n+1}$ and $b_0b_1 \ldots b_n \in \{0,1\}^{n+1}$, which can be viewed as polynomials $f(x) = a_nx^n + \cdots + a_1x + a_0$ and $g(x) = b_nx^n + \cdots + b_1x + b_0$. A deterministic protocol requires them to exchange all characters, an expensive solution. Alternatively the entities can follow the above randomised protocol:

- (Alice picks a prime $p \in [2n, 3n]$ and sends it to Bob).
- Alice randomly picks $x \in \{1, 2, \ldots, 2d\}$ and sends $x, f(x)$ to Bob.
- Bod computes $g(x)$ and compares it with $f(x)$. If the values match the algorithm decides that the polynomials (strings) are equivalent, otherwise that they are not.

Again the probability of success is at least $1/2$. In a communication protocol we seek to minimise the number of random bits used. As x requires $log_2 n$ bits our main concern is the value of $f(x)$ which we need to bound. In order to do this we take advantage of the fact they the PIT algorithm works for finite fields, like $Z_p = \{0, 1, \ldots, p-1\}$ where p is a large prime. With $f(x) \in Z_p$ and all operations done modulo p, we bound the size of $f(x)$ without affecting the number of roots.

Definition 3. Independence.

- *Events E_1, E_2 are independent if $Pr[E_1 \cap E_2] = Pr[E_1] \cdot Pr[E_2]$.*
- *Events $E_1, \ldots, E_k$ are mutually independent if for all $S \subseteq \{1, \ldots, k\}$ it holds that $Pr[\cap_{i \in S} E_i] = \prod_{i \in S} Pr[E_i]$.*

In order to enhance the success probability of the randomised PIT algorithm we pick $x_1, \ldots, x_k \in \{1, \ldots, 2d\}$ independently and uniformly at random. Then for all E_i about x_i we have that $Pr(E_1 \cap \cdots \cap E_k) = Pr[E_1] \ldots Pr[E_k]$, which implies that $Prob[x_1$ fails $\cap \cdots \cap x_k$ fails $] \leq (1/2)^k$, implying $Pr[success] \geq 1 - \frac{1}{2^k}$.

For this problem the deterministic algorithm always gives the correct answer requiring longer running time. The randomized algorithm may make mistakes but it is fast. The amplification through k time repetition of the algorithm reflects a trade-off between efficiency and correctness.

Definition 4. Conditional Probability.

- $Pr[E|F] = Pr[E \cap F]/Pr[F]$.
- $Pr[E \cap F] = Pr[F] \cdot Pr[E|F]$.

Lemma 5. Principle of deferred decision. *Let $E_1, \ldots, E_k$ be disjoint events s.t. $E_1 \cup \cdots \cup E_k = \Omega$. Then $Pr[F] = \sum_{i=1}^{k} Pr[E_i] \cdot Pr[F|E_i]$.*

Proof. (trivial) $Pr[F] = \sum_{i=1}^{k} Pr[F \cap E_i]$.

□

Example 6. *Let $b_1, \ldots, b_n \in \{0,1\}$. Then $Prob[b_i$ is even$] = 1/2$. We will use the principle of deferred decision to prove that $Pr[\sum b_i$ is even$] = 1/2$. $Pr[\sum b_i$ is even$] = \sum_{y_1,\ldots y_{n-1} \in \{0,1\}} Pr[b_i = y_i \ \forall i < n] \times Pr[\sum b_i$ is even $\mid b_i = y_i \ \forall i < n] = \sum_{y_1,\ldots y_{n-1}} Pr[b_i = y_i \ \forall i < n] \cdot \frac{1}{2} = \frac{1}{2}$.*

2.2 Multi-variate PIT

In this extension of the previous problem we have a polynomial of the form $f(x_1, x_2, \ldots, x_n) = x_1^2 x_2^3 x_3^5 + \ldots$ of total degree d, over reals for which we try to decide whether $f(x_1, \ldots, x_n) \equiv 0$ or not. In order to apply the single variable PIT randomised algorithm we may fix the value $n-1$ variables.

Lemma 7. Schwartz-Zippel Lemma. *If $S \subseteq \mathbb{R}$ then if we sample $x_1, \ldots, x_n$ from S independently and uniformly then $Pr[f(x_1, \ldots, x_n) = 0] \leq \frac{d}{|S|}$.*

Proof. (by Induction)

- Basis $n = 1$: the single variable case.
- Induction: Assume that the hypothesis holds for $n-1$.
- Let k be the largest power of x_1. Then we can write $f(x)$ as $f(x_1, \ldots, x_n) = \sum_{i=0}^{k} x_1^i \cdot q_i(x_2, \ldots, x_n)$, where $k \leq d$ and $q_k(x_2, \ldots, x_n) \not\equiv 0$.

Let $x = (x_1, \ldots, x_n)$. $Pr[f(x) = 0] = \sum_{y_2,\ldots,y_n \in S} Pr[x_i = y_i, \ \forall i > 1] \cdot Pr[f(x) = 0 \mid x_i = y_i \ \forall i > 1] = \sum_{y_2,\ldots,y_n \in S \& q_k(y_2,\ldots,y_n)=0} Pr[x_i = y_i] \cdot Pr[f(x) = 0 \mid x_i = y_i] + \sum_{y_2,\ldots,y_n \in S \& q_k(y_2,\ldots,y_n)\neq 0} Pr[x_i = y_i] \cdot Pr[f(x) = 0 \mid x_i = y_i]$.

Since q_k has degree $d-k$ we have from the inductive hypothesis that $\sum_{y_2,\ldots,y_n \in S \& q_k(y_2,\ldots,y_n)=0} Pr[x_i = y_i] \cdot Pr[f(x) = 0 \mid x_i = y_i] \leq \sum_{y_2,\ldots,y_n \in S \& q_k(y_2,\ldots,y_n)=0} Pr[x_i = y_i] \leq \frac{d-k}{|S|}$. Similarly, as $f(x)$ with $x_2, \ldots, x_n$ being fixed values has degree k, we have that $\sum_{y_2,\ldots,y_n \in S \& q_k(y_2,\ldots,y_n)\neq 0} Pr[x_i = y_i] \cdot Pr[f(x) = 0 \mid x_i = y_i] \leq \frac{k}{|S|} \cdot \sum_{y_2,\ldots,y_n \in S \& q_k(y_2,\ldots,y_n)\neq 0} Pr[x_i = y_i \ \forall i > 1] \leq \frac{k}{|S|}$.

Adding up the two bounds we obtain an overall upper bound of $\frac{d}{|S|}$ for $Prob[f(x) = 0]$. □

2.3 Bipartite perfect matching

Let $G = (V, E)$ be a bipartite graph in which we want to decide if a perfect matching exists. A perfect matching is a subset of the edge set E such that every vertex has exactly one edge incident on it. We define the $n \times n$ matrix $A_{u,v} = \begin{cases} x_{u,v}, & if\ \{u,v\} \in E \\ 0 & otherwise. \end{cases}$

Lemma 8. *$Det(A) \not\equiv 0$ if and only if G has a perfect matching.*

Based on this lemma we can use the PIT algorithm as an easy randomised algorithm for the bipartite perfect matching problem, with success probability at least 1/2:

- Randomly pick $x_{u,v} \in \{1, 2, \ldots, 2n\}$.
- Evaluate $Det(A)$.

3 Min-Cut

Let $G = (V, E)$ be a graph and $(S, V - S)$ be a partition of the graph's vertices. The cut of the partition is the set of edges between the the two sets $S, V - S$. We discuss a randomised algorithm for the Min-Cut problem.

We consider a merging algorithm $Merge(u, v)$ that merges vertices u, v into a new vertex, keeping parallel edges and deleting self loops.

MinCut Algorithm:

- Randomly pick an edge $(u, v) \in E$ and $Merge(u, v)$.
- Repeat until there are only two vertices left.
- Output the partition.

Theorem 9. *The MinCut algorithm has $Pr[success] \geq \frac{2}{n(n-1)}$*

Proof. Let $G = (V, E)$ be a graph and C be one min-cut of G with $|C| = k$.

Lemma 10. *If $|C| = k$ then all vertices have degree at least k and it follows that $|E| \geq nk/2$ (otherwise the partition would not be a min-cut).*

Lemma 11. *If G has a min-cut of size k then $G' = Merge(u, v)$ has a min-cut of size at least k for all u, v.*

Let C be a min-cut of G with $|C| = k$. Then our algorithm succeeds in finding the same min cut (denoted as $Pr[success]$) if throughout its execution all edges in C remain in G, having that $Pr[\text{success}] = Pr[\text{no merging is done over edges in } C]$. Let E_i be the event that the i-th edge picked for merging does not belong in C. The algorithm terminates with 2 vertices left, and thus requires $n - 2$ merging operations ($n = |V|$). Let F_i denote the event that the first i edges picked are not in C. Clearly we have that $Pr[E_1] = Pr[F_1] = \frac{|E|-k}{|E|} \geq 1 - 2/n$ (from Lemma **??**).

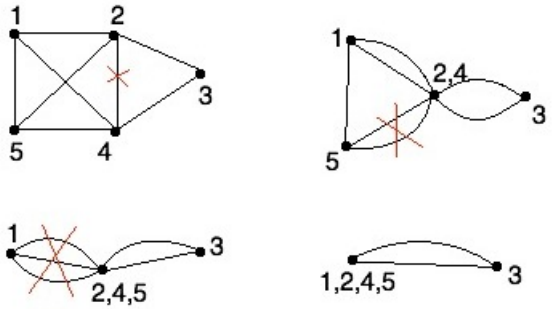

Figure 1: An example on the min-cut algorithm

Then we have that $Pr[F_2] = Pr[F_1 \cap E_2] = Pr[F_1] \cdot Pr[E_2|F_1]$ with $Pr[F_1] \geq 1-2/n$ and $Pr[E_2|F_1] = \frac{|E'|-k}{|E'|} = 1 - k/|E'| \geq 1 - \frac{k}{k(n-1)/2} = 1 - \frac{2}{n-1}$, with E' being the set of the edges in G'. Similarly we have that $Pr[E_i|f_{i-1}] \geq 1 - \frac{2}{n-(i-1)}$ We are interested in bounding $Pr[F_{n-2}] = Pr[E_{n-2} \cap F_{n-3}] = Pr[F_{n-3}] \cdot Pr[E_{n-2}|F_{n-3}] = Pr[F_{n-4}] \cdot Pr[E_{n-3}|F_{n-4}] \cdot Pr[E_{n-2}|F_{n-3}] \cdots \geq (\frac{n-2}{n})(\frac{n-3}{n-1}) \cdots \geq \frac{2}{n(n-1)}$. □

To enhance the success probability of the algorithm to $1/2$ we may repeat n^2 times, increasing the running time of the algorithm.