Randomness and Computation (INFR11089)

Lecture 1

Ilias Diakonikolas

January 13, 2015

Administrivia

Me: Ilias Diakonikolas ; email: ilias.d@ed.ac.uk

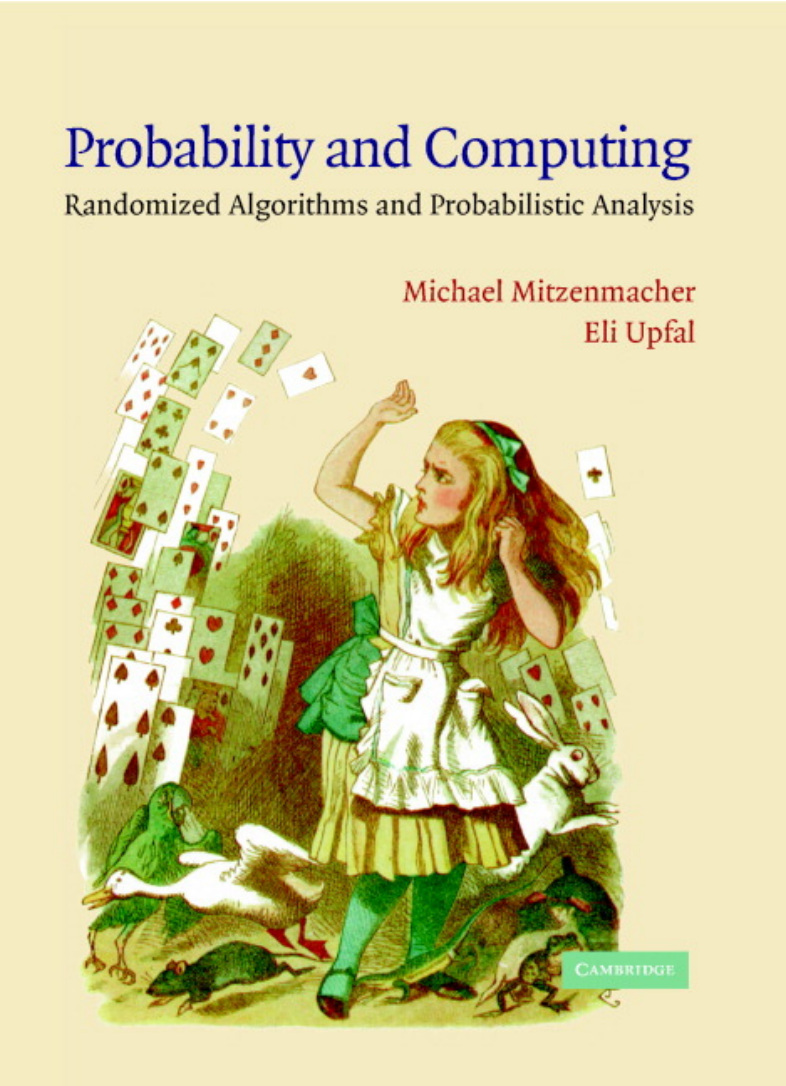
Office hours: IF-5.18, by appointment

Course Website:
http://www.inf.ed.ac.uk/teaching courses/rc/

Mailing list: rc-students@inf.ed.ac.uk

Evaluation: 2 Problem Sets (30%), Final Exam (70%)

Course Materials



- **Required Textbook:**
  Probability and Computing: Randomized Algorithms and Probabilistic Analysis,
  by Mitzenmacher and Upfal.

- **Course slides/notes (supplementary).**

- Useful online resources.

Prerequisites - Requirements

Good news: No formal prerequisites!

Recommended: Algorithms and Data Structures (**INFR09006)**

No programming!

Basic knowledge of discrete probability and algorithms:

- probability spaces and events, conditional probability and independence, random variables, expectations and moments, conditional expectation.
- asymptotic notation, basic sorting algorithms, basic graph algorithms.

Homework 0 (questionnaire about your background)

Probability and Computing

Why is probability (randomness) important for computing?

Applications (1)

- **Cryptography**.
- **Simulation.**
- **Statistics via Sampling.**
- **Learning Theory.**

Applications (2)

- **Queueing theory**.
- **Data Compression**.
- **Coding Theory**.
- **Data Structures.**

Applications (3)

- **Symmetry breaking**.
- **Theory of large networks**.
- **Quantum Computing**.
- **Statistics**.
- **Games and Gambling**.

Traits of Randomized Algorithms

- **Simplicity:** algorithms are often simple and elegant, e.g., randomized quick-sort. . .
- **Speed:** in many cases faster than the best known deterministic algorithms.

- Small probability of error
- May not terminate.

Verifying Polynomial Identities

- Given two polynomials of degree d, verify the identity

$$F(x) \equiv G(x)$$

- Naive algorithm:

1. Convert into canonical form, i.e., $\sum_{i=0}^{d} c_i x^i$
2. Check if the coefficients match.

Verifying Polynomial Identities

- Suppose that $F(x)$ is given to us as a product of monomials, i.e., $F(x) = \prod_{i=1}^{d} (x - a_i)$ and G in its canonical form.

- Transforming $F(x)$ in its canonical form in the obvious way requires $\Theta(d^2)$ multiplications.

- Can we do better?

Verifying Polynomial Identities

- Can we do better? Yes, by using **randomness**.

- Consider the following algorithm:

1. Choose an integer r uniformly at random from the set {1, …, 10d}.
2. Compute F(r) and G(r).
3. If F(r)≠G(r), output "NO". O/w, output "YES."

Running time: O(d).

Verifying Polynomial Identities

1. Choose an integer r uniformly at random from {1, …, 10d}.
2. Compute F(r) and G(r).
3. If F(r)≠G(r), output "NO". O/w, output "YES."

Analysis:

Suppose $F(x) \equiv G(x)$. Then the algorithm is always correct.

Suppose $F(x) \neq G(x)$. Then the algorithm might give the wrong answer. What is the probability of this **event**?

Verifying Polynomial Identities

1. Choose an integer r uniformly at random from {1, …, 10d}.
2. Compute F(r) and G(r).
3. If F(r)≠G(r), output "NO". O/w, output "YES."

Analysis:

Suppose $F(x) \neq G(x)$. Then the algorithm might give the wrong answer. What is the probability of this **event** E ?

Sample space: $\Omega = \{1,\ldots 10d\}$

Error event: $E = \{r \in \Omega : F(r) = G(r)\}$

What is the probability of this event, i.e., $\Pr(E)$?

Verifying Polynomial Identities

1. Choose an integer r uniformly at random from {1, …, 10d}.
2. Compute F(r) and G(r).
3. If F(r)≠G(r), output "NO". O/w, output "YES."

Analysis:

Error event: $E = \{r \in \Omega : F(r) = G(r)\}$; $\Pr(E)$?

- F-G is a degree d polynomial (at most d roots). Hence, $|E| \le d$.

- Since all values of r have the same probability

$$\Pr(E) = |E| \cdot (1/10d) \le 1/10$$

Verifying Polynomial Identities

1. Choose an integer r uniformly at random from {1, …, 10d}.
2. Compute F(r) and G(r).
3. If F(r)≠G(r), output "NO". O/w, output "YES."

Suppose we want to make the error probability even smaller.

Repeat algorithm k times.

By independence probability of error becomes 10^{-k}.

Running time: O(kd).

Verifying Matrix Multiplication

- Given three nxn matrices A, B, C (over GF(2)) verify whether

$$AB = C$$

- Naive algorithm:

1. Multiply A and B together.
2. Compare result to C.

Verifying Matrix Multiplication

1. Multiply A and B together.
2. Compare result to C.

Running time?

Matrix multiplication (very well-studied).

- "Obvious" algorithm: $O(n^3)$.
- [Strassen, 1969]: $O(n^{2.81})$.
- …
- [Coppersmith – Winograd, 1987]: $O(n^{2.376})$.
- …
- [Stothers, 2010]: $O(n^{2.3737})$.
- [Vassilevska-Williams, 2012]: $O(n^{2.3727})$.

Verifying Matrix Multiplication

Given three n x n matrices A, B, C verify whether $A\,B = C$

Best known deterministic algorithm: $\Omega(n^{2.3727})$..

Can we do better? Yes, by using randomization.

Algorithm (Rusins Freivalds, 1979):

1. Choose $\bar{r} = (r_1, \ldots r_n) \in \{0,1\}^n$ uniformly at random.
2. Compute $y = (AB) \cdot \bar{r}$ and $z = C \cdot \bar{r}$.
3. If $y \neq z$, output "NO". O/w, output "YES."

Verifying Matrix Multiplication

Algorithm (Rusins Freivalds, 1979):

1. Choose $\bar{r} = (r_1, \ldots r_n) \in \{0,1\}^n$ uniformly at random.
2. Compute $y = (AB) \cdot \bar{r}$ and $z = C \cdot \bar{r}$.
3. If $y \neq z$, output "NO". O/w, output "YES."

Running time: $\Theta(n^2)$.

Analysis:

- If $A\,B = C$ the algorithm is always correct.
- O/w, the algorithm may give the wrong answer.

  What is the error probability?

Verifying Matrix Multiplication

Algorithm (Rusins Freivalds, 1979):

1. Choose $\bar{r} = (r_1, \ldots r_n) \in \{0,1\}^n$ uniformly at random.
2. Compute $y = (AB) \cdot \bar{r}$ and $z = C \cdot \bar{r}$.
3. If $y \neq z$, output "NO". O/w, output "YES."

Analysis:

- Suppose $A\,B \neq C$. The error event is $E = \left\{ \bar{r} \in \{0,1\}^n : AB\bar{r} = C\bar{r} \right\}$

Claim: $\Pr(E) \leq 1/2$

Proof: By assumption $D = A\,B - C$ has at least one non-zero entry.

Assume wlog that $d_{11} \neq 0$. We have that

$$r_1 = -\sum_{j=2}^{n} d_{1j} r_j \Big/ d_{11}$$

Verifying Matrix Multiplication

Algorithm (Rusins Freivalds, 1979):

1. Choose $\bar{r} = (r_1, \ldots r_n) \in \{0,1\}^n$ uniformly at random.
2. Compute $y = (AB) \cdot \bar{r}$ and $z = C \cdot \bar{r}$.
3. If $y \neq z$, output "NO". O/w, output "YES."

Claim: $\Pr(E) \leq 1/2$

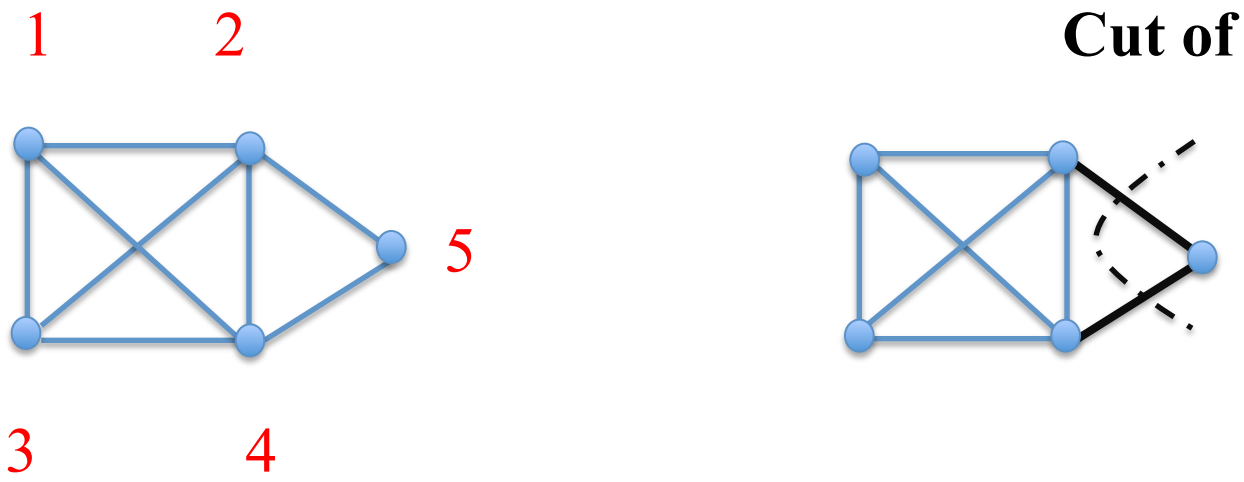
Proof: For $D = A\,B - C$ with $d_{11} \neq 0$ we have $r_1 = -\sum_{j=2}^{n} d_{1j} r_j \Big/ d_{11}$

- Since the r_j's are independent, for a fixed setting of $r_2, \ldots, r_n$, the RHS is fixed.
- Hence, the **conditional probability** that r_1 is equal to this value is at most ½.
- Since this holds for every setting of of $r_2, \ldots, r_n$ the claim follows. □

Finding a Minimum cut in a Graph

Let G = (V, E) be an undirected graph.

- Cut = set of edges whose removal makes the graph disconnected.



- **Size** of a cut = number of edges it contains
- **Minimum** cut = cut of minimum size

Finding a Minimum cut in a Graph

Minimum cut problem: Given G compute a minimum cut.

Let n = number of vertices; m = number of edges.

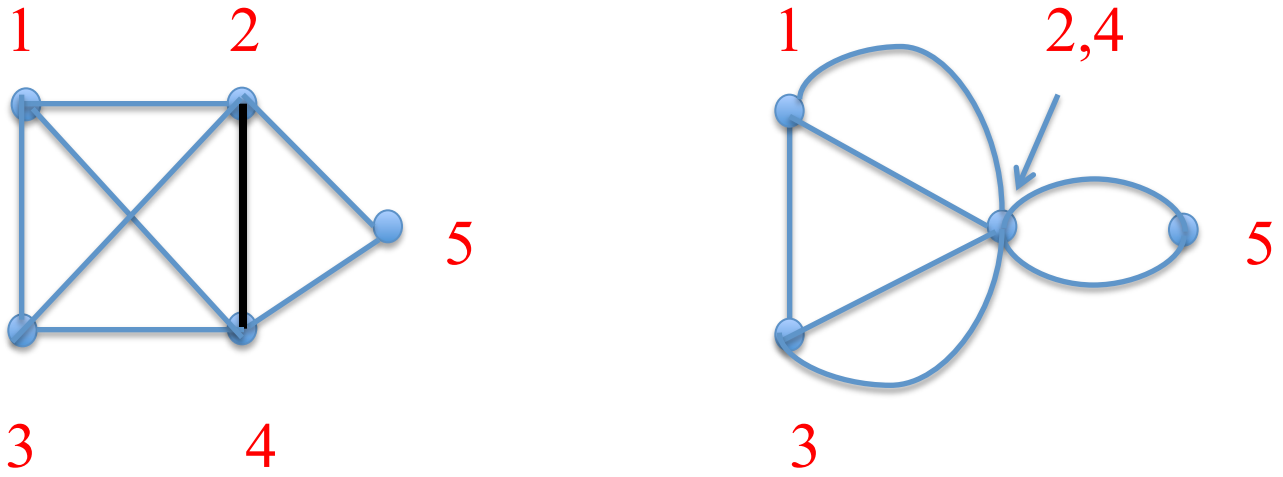
- Fastest known deterministic algorithms: $\Omega\,(mn)$
- Fastest known randomized algorithm: $O\,(n^2 \log^3 n)$
  [Karger'93; Karger-Stein'96]

Edge Contraction Operation

Contraction of an edge (u, v):

- Merge the two vertices into one.
- Eliminate all edges between u and v.
- Keep all other edges.

Example:



Random Contraction Algorithm

Random Contraction Algorithm [Karger'93]:

Repeat

- Choose an edge (u,v) uniformly at random from E.
- Contract the vertices u and v to a super-vertex $w = \{u,v\}$.
- Keep parallel edges but remove self-loops.

until G has only 2 vertices.

- Report the corresponding cut.

Intuition: if min cut C is small, probability we choose edge in C also small.

Lemma: The algorithm outputs a min-cut with probability at least $2/n^2$.

Random Contraction Algorithm

Repeat

- Choose an edge (u,v) uniformly at random from E.
- Contract the vertices u and v to a super-vertex $w = \{u,v\}$.
- Keep parallel edges but remove self-loops.

until G has only 2 vertices.

- Report the corresponding cut.

Lemma: The algorithm outputs a min-cut with probability at least $2/n^2$.

Proof Sketch: Fix a minimum cut C.

Let E_i be the event: " edge contracted in iteration i not in C ."

Want $\Pr\left(\bigcap_{i=1}^{n-2} E_i\right) \geq \frac{1}{n^2}$. Can show that $\Pr\left(E_i \middle| \bigcap_{j=1}^{i-1} E_j\right) \geq 1 - \frac{2}{n-i+1}$.

Hence,

$$\Pr\left(\bigcap_{i=1}^{n-2} E_i\right) = \prod_{i=1}^{n-2} \Pr\left(E_i \middle| \bigcap_{j=1}^{i-1} E_j\right) \geq \frac{2}{n(n-1)}$$

□

Conclusions

- Reading material: Chapter 1 of textbook.

- Check out class webpage.