CS 221 Computational Complexity, Lecture 11 Feb 27, 2018

Approximate Counting and Parity

Instructor: Madhu Sudan *Scribe: Andrei Ciupan*

1 Goals for today & Reminders

1. Today we will:
   (a) Introduce approximate counting
   (b) Introduce the theorem Parity $\notin AC^0$
2. Office hours today 5-6PM
3. Pset 3 due this Friday

2 Introduction

Last time we introduced #**P** and Permanent. In the following lectures we will be using counting arguments for our results. Today we focus on approximate counting and Parity.

3 Approximate counting

Definition 1. *Consider a function $f : \{0,1\}^* \mapsto \mathbb{R}^{\geq 0}$. A function $A : \{0,1\}^* \mapsto \mathbb{R}^{\geq 0}$ is an α* - approximation algorithm to f *if for all x of length n, we have*

$$\frac{f(x)}{\alpha(n)} \leq A(x) \leq f(x) \cdot \alpha(n).$$

Definition 2. *An approximation scheme for f is a collection of approximation algorithms such that for all $\varepsilon > 0$ there exists a $(1+\varepsilon)$ approximation algorithm for f in polynomial time.*

A stronger notion is the following:

Definition 3. *A Fully Polynomial Randomized Approximation Scheme is an approximation scheme composed of one algorithm such that for all $\varepsilon > 0$, it produces a $(1+\varepsilon)$ approximation of f and runs in time poly $\left(\frac{n}{\varepsilon}\right)$ for inputs of length n.*

Note many problems do not admit an approximation scheme:

1. An approximation to $\#SAT$ determines whether the number of satisfying assignments is greater than zero
2. An approximation to $\#Cliques(G)$ (the total number of cliques in a graph G) solves the problem of counting the size of the largest clique in a graph. For a given graph G of n vertices, consider the graph $G \times K_n$ defined as follows:
   (a) $V(G \times K_n) = V(G) \times V(G)$ (the convention is that $V(G')$ represents the set of vertices in a graph G'.
   (b) There exists an edge between (v, w) and (v', w') in $G \times K_n$ if and only if either $v = v'$ or there exists an edge between v and v' in G.

Then if $\omega(G)$ is the size of the maximal clique in G we have the following result:

$$2^{n\cdot\omega(G)} \leq \#\text{ cliques in } G \times K_n \leq 2^{n\omega(G)} \cdot \binom{n}{2}^{\omega(G)}.$$

This implies that a $\frac{2^n}{n^{\omega(G)}}$ - approximatin algorithm to $\#Clique(G \times K_n)$ determines $\omega(G)$.

Exercise 1. Prove the two claims above. The following results will be useful:

1. The number of cliques in a graph with n vertices and maximum clique size k is between 2^k and n^k.
2. $\{(v_1, w_1), \ldots, (v_q, w_q)\}$ is a clique in $G \times K_n$ if and only if $\{v_1, \ldots, v_q\}$ is a clique in G.

Madhu also mentions that these kinds of approximate counting problems have also enjoyed a long history in the field of Monte Carlo algorithms. These also have complexity theoretic interpretations. Still there is a collection of nice results coming from approximations.

Jerrum - Valiant - Vazirani prove a generalization of the following result:

Theorem 4. *There exists an approximate algorithm for counting the number of matchings in a graph G* $\Longleftrightarrow$ *there exists an approximate algorithm for uniformly sampling a matching in G.*

Proof. Assuming a $(1+\varepsilon)$ approximate counting algorithm $\#Match(G)$ for the number of matchings in a graph G, we construct a random sampling from the set of matchings in G as follows:

1. Pick a random edge $e = (u, v) \in G$. With probability $\frac{\#Match(G - \{u,v\})}{\#Match(G)}$ select e. If an edge was selected, go to step 2. Otherwise, repeat.
2. Let $e = (u, v)$ be the most recently chosen edge. Repeat step 1 above for $G \leftarrow G - \{u, v\}$, until we run out of vertices.
3. Output the set of edges chosen.

The output of this algorithm gives a $(1+\varepsilon)^{O(n)}$ approximation for random sampling.

For the reverse implication we "invert" the previous process:

Pick an edge $e = (u, v)$ from G. Run the uniform sampler to estimate p, the probability that edge e belongs to a matching of G.

1. If $p < \frac{1}{2}$, recursively find the number of matchings in $G - e$ and multiply it by $\frac{1}{1-p}$. Note that in the graph $G - e$ we just remove edge e, not the endpoints.
2. If $p \geq \frac{1}{2}$, recursively find the number of matchings in $G - \{u, v\}$ and multiply it by $\frac{1}{p}$. Note that in the graph $G - \{u, v\}$ we remove edge e and the endpoints.

□

Exercise 2. Prove (hint: by induction) that in the second implication above, we obtain an error of $(1+\varepsilon)^{O(\#\text{ edges in G})}$ if the error of the sampler is $1+\varepsilon$.

Exercise 3. Prove that approximating the number of cycles in a graph is NP-hard.

Hint: reduce from Hamiltonian Cycle.

Now we move on to the second part of today's class:

4 Approximate counting and known complexity classes

Exercise 4. If PH has a complete problem, then it collapses.

Theorem 5. *Approximate counting is in Σ_3^P.*

Proof. We will show that given a boolean formula ϕ, distinguishing between whether the number of satisfying assignments is $\geq 2^{k+1}$ and $\leq 2^k$ is in Σ_3^P.

The idea is to find a collection of hash functions $\{h_i\}$ which satisfy pairwise independence, such that if the number of satisfying assignments is "small", then the number of satisfying assignments hashed to zero is small, and if the number of satisfying assignments is large, then the number of satisfying assignments hashed to zero is also large.

Exercise 5. Complete this proof. Hint : it is in the same spirit as the proof from lecture 10. □

Remark Just like we saw the amplification argument in the previous lectures, we can see a similar one here: in order to increase the number of satisfying assignments of a CNF formula, one can consider the "squared" formula on twice the number of variables, which simply repeats the initial formula twice. The number of satisfying assignments to this formula is the square of the number of satisfying assignments of the original one. This allows us to amplify the gap in the number of satisfying assignments, if needed (for example for a result like the theorem above).

Theorem 6. *Toda's theorem : $PH \subseteq \#P$ (we will revisit this later in the course)*

Theorem 7. *There exists an oracle A such that $PH^A \neq \#P^A$*

Theorem 8. *Parity $\notin AC^0$.*

Proof. We prove that if a circuit of depth d computes parity for an input of length n, then its size is at least $2^{\frac{1}{2}n^{1/2d}}$.

The idea is to prove that any proper polynomial of degree $\sqrt{n}$ in $\mathbb{F}_3$ differs from parity on at least $\dfrac{2^n}{50}$ inputs, while there exists a proper polynomial of degree $\sqrt{n}$ that differs from any circuit of size s and depth d on at most $2^{n-\frac{1}{2}\cdot n^{1/d}}$ inputs.

Combining these two gives the result as it implies that the size of the circuit can't be polynomial. □