3.8 Approximate counting vs. approximate sampling

We assume that we have access to a uniform distribution on $\{0,1\}^n$, and wish to obtain a uniform distribution on some other sets. Some probability distributions of interest are:

1. Uniform distribution on permutations of the set $[n]$. (These can be obtained by straightforward sampling, or by shuffling techniques.)

2. Uniform distribution on the spanning trees of an undirected graph. (Very nice methods here: matrix tree theorem, Broder, Aldous, loop-erased random walk.)

3. Uniform distribution on solutions to a knapsack problem, where a knapsack problem is defined as:
Given $a_1, a_2, ..., a_n, b \in \mathbb{N}$, a solution is a vector $v = (v_1, v_2, ..., v_n) \in \{0,1\}^n$ s.t. $\sum_i v_i a_i \leq b$

4. Uniform distribution on matchings, or perfect matchings, of a graph.

5. Uniform distribution on self-avoiding walks of length n in $\mathbb{Z}^d$.

For most such problems we have to settle for approximate sampling. (Spanning trees is an interesting exception!) What we'll see here is that this is intimately related to the more usual question in complexity theory of *counting* combinatorial objects.

To set the stage, suppose we have a polynomial-time predicate ϕ , where $\phi : \{0,1\}^n \to \{0,1\}$, and we want to either count the number of solutions to ϕ (i.e., $|\phi^{-1}(1)|$) or sample uniformly from the solutions to ϕ.

As discussed earlier, doing either task perfectly is often computationally hard. For example if ϕ is a Boolean formula, then we are asking for access (in either form) to the satisfying assignments, which is at least as hard as asking whether the formula is satisfiable. However, as we will see later, in many situations where exact solutions are hard, an approximate solution can be obtained in polynomial time. Specifically, we we will be interested in multiplicative $1 \pm \varepsilon$ approximations of a number $0 \leq \theta \leq 1$ (such as the fraction of assignments which are satisfying). This is more meaningful than additive $\pm\varepsilon$ approximation if the solution-space that we are interested in represents an exponentially small fraction of the total sample space, where a multiplicative approximation gives us an upper limit on our percent error, while an additive approximation does not accomplish that.

Note: Since a poly-time algorithm for evaluating ϕ exists, one possibility is to do rejection sampling—i.e., generate a random vector x in $\{0,1\}^n$, check if $\phi(x) = 1$ and if so return x, and otherwise repeat. As mentioned earlier, this is unbiased (as an estimator for the fraction of satisfying assignments) but the variance $\theta(1-\theta)$ is large. (We'd need variance $< \theta^2$ for multiplicative approximation.)

So unless we say otherwise, we'll be interested in a FPRAS. For example, we've seen a FPRAS for DNF formulas, but we note not to expect one for CNF formulas since the existence problem ("does this formula have a satisfying assignment") is NP-complete for CNF formulas.

We're now interested in relating the counting task to the task of approximate sampling. A perfect sampling procedure would enable us to sample from the distribution p that is exactly uniform on $\phi^{-1}(1)$; in approximate sampling we're willing to sample from a distribution q that is, in L_1 distance, within ε of p (so $||p-q||_1 = \sum_x |p(x) - q(x)| \leq \epsilon$). So, we have two computational tasks, counting and sampling, in both of which we've allowed some "distortion"; we're interested in this lecture in relating the complexities of these two tasks.

Rough definition:[4] A search/sampling/counting problem is self-reducible if $\exists$ a representation of n bits for the objects defined by problem P s.t. for $i = 0, 1$, the set of objects whose first bit is i are the objects of another problem P_i of size $\leq |P|$ and P_i is polynomial-time computable from P.

Most problems we encounter in practice are self-reducible. [5] Examples of self-reducible problems:

1. SAT: P is a boolean formula, objects are the satisfying assignments, $|P|$ is the formula size, n is the number of variables.

2. Set of matchings of an undirected graph: P is the graph, objects are the matchings, n is the number of edges, restriction is to choose an edge e between vertices u and v, P_0 is $P - e$, P_1 is $P - \{u, v\}$.

Theorem 39 (Jerrum, L. Valiant, V. Vazirani [26])**.** *TFAE for a self-reducible problem:*

1. $\forall k$ $\exists$ *a poly-time algorithm that samples from a distribution* p *on* $\phi^{-1}(1)$ *s.t.* $||p - uniform||_1 \leq n^{-k}$. *("$L_1$-approximate sampling")*
2. $\forall k$ $\exists$ *a poly-time algorithm that, with probability* $\geq 1 - e^{-kn}$, *estimates* $|\phi^{-1}(1)|$ *within factor* $1 \pm n^{-k}$. *("Approximate Counting")*
3. $\forall k$ $\exists$ *a poly-time algorithm that samples from a distribution* p *on* $\phi^{-1}(1)$ *s.t. for all* x, $|\log(p(x)|\phi^{-1}(1)|)| \leq n^{-k}$. *("Pointwise approximate sampling")*

It might look unreasonable that we can go from 1 to the seemingly stronger 3, but this is possible because of the assumption that we can achieve the required approximations anywhere in the self-reducibility tree.

In the proof we freely move between bounds of the form e^x and $1 + x$, since for x near 0, inequalities hold by just paying a factor of 2 in x.

Proof. It is clear that $3 \to 1$.

1 (L_1 Sampling) $\to$ 2 (Counting).

First note that it is sufficient to succeed with probability $2/3$; then finish off with the median trick.

Consider any root to leaf path $a_0, a_1, \ldots, a_{n'}$ in the self-reducibility tree. (Here $n' \leq n$ but may depend upon the particular leaf.) For $i \geq 1$ let r_i be the fraction of leaves under a_{i-1} that are under

[4] There are several different definitions of self-reducibility, which differ slightly; some are more useful for algorithms and complexity, some for cryptography. One version serves our purpose, from Meyer and Paterson [42] (who in turn followed earlier work by Schnorr [52]), is

Definition 37. *Let $|x|$ denote the length of the string x. A partial order $<$ on $\{0,1\}^*$ is "polynomially well-founded and length-related", OK for short, if*

- *(a) There is a polynomial p such that every finite $<$-decreasing chain is shorter than p of the length of its maximum element; and*
- *(b) $x < y$ implies $|x| \leq p(|y|)$ for some polynomial p, and all $x, y \in \{0,1\}^*$.*

Obviously choosing to define $x < y$ iff $|x| < |y|$ is OK.

Definition 38. *A language $L \subseteq \{0,1\}^*$ is self-reducible iff there is an oracle Turing machine, M, and an OK partial order on $\{0,1\}^*$ such that, given an oracle for L, M accepts L in polynomial time, and moreover, on any input $x \in \{0,1\}^*$, M asks its oracle only about words strictly less than x in the partial order.*

[5] Quoting [26]: "Very many naturally occurring relations are self-reducible; examples include 1-factors in an undirected graph and satisfying assignments to a CNF- or DNF-formula. In fact, problems which cannot be formulated in a self-reducible way seem to be the exception rather than the rule.

E.g., most known problems in NP are self reducible, and under some definitions, all problems in P do. However, some limits of these assertions were clarified later literature. For example, the problem of coloring a planar graph with 4 colors (notice this is not a decision problem) is, providing P $\neq$ NP, not self-reducible under the Schnorr definition [36, 22].

a_i. The total number of leaves in the tree is equal to (along *any* path)

$$|\phi^{-1}(1)| = \prod_{i=1}^{n'} \frac{1}{r_i}$$

This suggests the following algorithm: Sample from the problem at node a_{i-1} until the fractions of leaves in its children can be estimated to within additive $n^{-k-1}/4$ with probability of error $1/3n$. Then proceed to the child in which this estimate q_i is at least $1/2$ (recall that the tree is binary). Unless an error has occurred, $e^{-n^{-k-1}} \leq \frac{q_i}{r_i} \leq e^{n^{-k-1}}$. After finally reaching a leaf, set estimator

$$T := \prod_{i=1}^{n'} \frac{1}{q_i}.$$

Unless an error event has happened somewhere (by a union bound this probability is $\leq 1/3$),

$$\frac{|\phi^{-1}(1)|}{T} = \prod_{i=1}^{n'} \frac{q_i}{r_i} \in [e^{-n^{-k}}, e^{n^{-k}}]$$

(where this product is along the path chosen by the algorithm).

2 (Counting) $\to$ 3 (Pointwise Sampling).

Consider an internal node of the tree, and let q, q_0 and q_1 be the true fractions of leaves in the corresponding subtrees; if we are at the parent node, we use the counting algorithm to obtain estimates p, p_0 and p_1, so that with probability $\geq 1 - 1/3n$, all these estimates are within factor $e^{-n^{-k-1}/4}$ of the truth (i.e., the q values). Now, it follows that $|\log \frac{p_i}{p} \frac{q}{q_i}| \leq n^{-k-1}$. Adding this over levels, we have that at any leaf v, $|\log \frac{p_v}{q_v}| \leq n^{-k}$. But of course $q_v = \frac{1}{|\phi^{-1}(1)|}$.

□