Uniform Generation & Approximate Counting

Given graph $G$:

Tasks:
- output a perfect matching
- output a uniformly chosen perfect matching
  - e.g. Let $M_6 = \pi_1 M_4$ is a perfect matching in $G_3$
- count number of perfect matchings
- output $\gamma \in M_6$
- output a spanning tree
- output a uniformly chosen spanning tree
  - e.g. Let $S_6 = \pi_1 S_4$ is a spanning tree of $G_3$
- count # of spanning trees

Given Boolean formula $\phi$ (DNF, CNF?)

Tasks:
- output sat assignment
- output uniformly chosen sat assignment
- count # of sat assignments

Complexities:

$\#P$-complete: Note if can count $\#\text{SAT}$ can solve SAT so $\#\text{SAT}$ is at least as hard.

What about $\#\text{DNF}$?

Transform $\phi$ in CNF to $\not\phi$ in DNF

$\#\phi \equiv 2^{\#\not\phi}$

So $\#\text{DNF}$ is also $\#P$-complete
<table>
<thead>
<tr>
<th>Decision problem</th>
<th>Counting problem</th>
<th>approx counting problem</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNF</td>
<td>NP-complete</td>
<td>hard</td>
</tr>
<tr>
<td>DNF</td>
<td>poly time</td>
<td>poly time (last lecture)</td>
</tr>
<tr>
<td>Matching</td>
<td>poly time</td>
<td>poly time</td>
</tr>
<tr>
<td>Spanning graphs</td>
<td>poly time</td>
<td>polytime</td>
</tr>
<tr>
<td>your favorite problem</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>
Approximate Counting

Given \( \phi \)

subject to \( \# \text{sat assignments to } \phi \)

Output \( y \) subject to

\[
\frac{z}{1+\epsilon} \leq y \leq z \cdot (1+\epsilon)
\]

with probability \( \geq \frac{3}{4} \)

Hope runtime polynomial in \( |\phi|, \frac{1}{\epsilon} \)

Note: problem 4 on HW 4: such an algorithm can be used to give a \textit{poly} time \textit{Valg} for SAT.

Question: can we get \textit{fpras} for \#SAT?

Answer: no if \( \text{BPP} \neq \text{NP} \)

\( \text{fpras for } \#\text{SAT} \Rightarrow \text{poly time } \text{Valg for SAT} \)

\[
\text{if } \phi \text{ SAT then } \#\phi \geq 1 \Rightarrow y \geq \frac{1}{1+\epsilon}
\]

\[
\text{if } \phi \text{ UNSAT then } \#\phi = 0 \Rightarrow y = 0
\]

Next Question: can we get \textit{fpras} for \#DNF?
Answer: Yes!!
will use
(1) uniform generation of DNF assignments
(2) "Downward self-reducibility" of DNF:

Downward Self-reducibility (dsr)
Can compute problem solution by
solving problem on smaller subproblems
+ putting together answers via polynomial computation.

Why is A - DNF dsr?

\[ \# \phi(x_1, x_2, \ldots, x_n) = \# \phi(x_1 = T_1, x_2, \ldots, x_n) + \# \phi(x_1 = T_2, x_2, \ldots, x_n) \]
both are # of DNFs
but in n-1 vars

E.g.

\[ \# (x_1 \overline{x_2} \vee x_1 x_2 x_3 \vee \overline{x_2} \overline{x_3}) = \# (\overline{x_2} \vee x_2 x_3 \vee \overline{x_2} \overline{x_3}) \]
+ \# (\overline{x_2} \overline{x_3})
both are DNFs in
n-1 vars

Count # of settings
of \( x_1 x_3 \)
that satisfy (not \( x_1 x_2 \)).
Downward self-reducibility tree.

Let \( F_{b_1 b_2 \ldots b_n} \equiv \# \phi (x_1 = b_1, x_2 = b_2, \ldots, x_n = b_n) \).

Each node's value is the sum of its children.

Leaves are either true or false.

Example:
\[ \# (\overline{x_1} \overline{x_2} \lor x_1 x_2 x_3 \lor \overline{x_2} \overline{x_3}) \]
Approximate Counting algorithm for #ONE:

Let \( S_1 = F_1/F \)

\[
\downarrow
\]

\[
F = F_1 \frac{1}{S_1}
\]

Can estimate via sampling:
- Uniformly generate \( K \) SAT assignments \( \{x_i\} \)
- \( \tilde{S}_1 \leftarrow \# \) generated assignments in which \( x_i = T \)

Can do this from \( S_0 + F_0 \) as well.

To compute \( F \):
- estimate \( S_1 \) via sampling uniformly
- Recursively compute \( F_1 \)

Continuing: if \( S_{b_1 \ldots b_x} = \frac{F_{b_1 \ldots b_x}}{F_{b_1 \ldots b_{x-1}}} \)

then \( \frac{F_{b_1 \ldots b_{x-1}}}{S_{b_1 \ldots b_x}} = \frac{F_{b_1 \ldots b_x}}{S_{b_1 \ldots b_x}} \) ← estimate

\( S_b, F = \frac{F_{b_i}}{S_{b_i}} \)

\[
= \frac{F_{b_i b_2}}{S_{b_i b_2}} = \frac{F_{b_i b_2 b_3}}{S_{b_i b_2 b_3}} = \ldots = 1
\]

Problems:
1. what if \( S_{b_i \ldots b_x} = 0 \)?
2. we only approximate \( S_{b_i \ldots b_x} \)
Problem 1:

What if \( S_{b_1 \ldots b_k} = 0 \)?

**idea**: go down to "larger" child

(since sampling might guess wrong when picking larger child but this only happens when both children have lots of sat assignments)

Claim if always pick \( b_i \) st. \( F_{b_1 \ldots b_i} > F_{b_1 \ldots b_{i-1}} \)
then reach a satisfying assignment leaf.

Problem 2:

**only get additive estimates**

**idea**: estimate each \( S_{b_1 \ldots b_k} \) to within \( \left( 1 + \frac{\varepsilon}{2^n} \right) \)

\[ \frac{\varepsilon + \varepsilon}{2^n} \leq r \left( 1 + \frac{\varepsilon}{2^n} \right) \]

\[ \leq r \left( 1 + \frac{\varepsilon}{2^n} \right) \]

**Claim**: output \( \leq F_{b_1} \leq F_{b_1 b_2} \leq \cdots \leq \frac{1}{\prod S_{b_1 \ldots b_k}} \]

\[ \leq \left( 1 + \frac{\varepsilon}{2^n} \right)^n = F \cdot \left( 1 + \frac{\varepsilon}{2^n} \right)^n \leq F(1 + \varepsilon) \]

Similarly, output \( \geq \frac{F}{(1 + \varepsilon)} \)
Algorithm to estimate $\#DNF$:

- estimate $S_0, S_1$ using uniformly generated sat assignments
- let $b_1 \leftarrow \arg \max_{b_1} \mathbb{E}_{b_0, S_1, S_2}$
- Recurse on $F_{b_1}$

Runtime?

$n \cdot $ samples required to get $\frac{E}{4n}$ additive error $\cdot$ runtime of uniform generator $\uparrow$

$\text{poly in } (\frac{E}{4n})^{-1}$ via Chernoff bounds $\uparrow$

$\text{poly in } n$

$$= \text{poly} \left( \frac{1}{E}, n \right)$$

$$\Pr \left[ \text{algorithm works} \right] = \Pr \left[ \text{estimate falls within } \frac{E}{4n} \text{ additive error at each of } n \text{ calls} \right]$$

$$\geq 1 - n \cdot \Pr \left[ \text{estimate bad in single call} \right]$$

- Chernoff bounds

This works for any $d$sr problem

polytime (almost)-uniform generation of solns to $\# 3$ $\Rightarrow$ polytime approximate counting of solns to $\# 3$

("$?$" needs to be $d$sr)

What about $\leq ?$
Uniform Generation + Approx Uniform generation

\[ \text{def. Uniform generator for solns of problem T on input } X, \text{ eg. assignment, eg. boolean formula} \]

\[ \text{Define set of solns } S_x = \{ z \mid z \text{ is a soln to } x \} \]

\[ \text{outputs y uniformly from } S_x \]

\[ \forall y \in S_x \quad \Pr[\text{output } y] = \frac{1}{|S_x|} \]

\[ \text{do not output } y \notin S_x \]

\[ \text{runs in time } \text{poly}(|x|) \]

\[ \text{def almost uniform generator as above, but has extra input } \varepsilon \]

\[ \forall y \in S_x \quad \frac{1}{|S_x|}, \frac{1}{1+\varepsilon} \leq \Pr[\text{output } y] \leq \frac{1}{|S_x|} \cdot (1+\varepsilon) \]

\[ \text{runs in time } \text{poly}(|x|, \frac{1}{\varepsilon}) \]
Approximate Uniform generation from approximate counting algorithms:

Assume (perfect) counting alg for \#DNF

Recurrence Algorithm: At \( b_0 \ldots b_i \)

1. Use (perfect) counter to compute
   \[
   r_0 = \frac{F}{b_i} + \frac{F}{b_i} \cdot b_0 \cdot r_1
   \]
   Go down left branch with prob \( \frac{r_0}{r_0 + r_1} \)
   and right branch o.w.

Claim (1) always reach a sat assignment

(a) \( \Pr \left[ \text{output assignment } b = (b_0 \ldots b_n) \right] = \frac{F}{b_i} \cdot \frac{F}{b_i} \cdot \frac{F}{b_i} \cdot \ldots \cdot \frac{1}{F} = \frac{1}{F} \)

which is same for each sat assignment.

\( \Rightarrow \) Uniformly generate a sat assignment

Question: What if only approx counter?

Answer:

RHS \leq \frac{1}{F} \cdot \left( \frac{1 + \varepsilon}{1 - \varepsilon} \right)^n \leq \frac{1}{F} \cdot \left( \frac{1 - \varepsilon}{1 - \varepsilon} \right)^n \leq \frac{1}{F} \cdot \left( 1 + \frac{\varepsilon}{2n} \right)

\( \Rightarrow \) close to uniform generation of sat assignment

This also works for any DSR problem!

Theorem (Valiant-Vazirani) for any problem in NP that is DSR

Approx counting #sols doable in Poly time iff Almost Uniform generation doable in Poly time

(Almost all we know are)