Lecture 5

Uniform generation

- Uniformly generating satisfying assignments to DNF formula

Counting problems

- #P

Approximate Counting

- Connection to uniform generation

Uniform sampling of satisfying assignments to DNF formula

DNF Formula:

"or of ands"

e.g. $\phi(x_1 \cdots x_n) = x_1\bar{x}_2x_3 \vee x_2\bar{x}_3x_4x_{10} \vee x_8\bar{x}_{10}x_{11} \vee \ldots$

Notation: implicit $\wedge$'s (we don't bother to write them)

Task: Find satisfying assignment to ϕ

easy!

pick one term + set literals in it to true

(satisfied if $\exists$ term s.t. not both x_i + $\bar{x}_i$ in it)

Task: Find random satisfying assignment to ϕ

↑ uniform over all sat assignments

Is it doable???

Special case:

Only one conjunction

$$F = y_1 \wedge y_2 \wedge \ldots \wedge y_k \quad \text{for } y_i \in \{x_1, \bar{x}_1, x_2, \bar{x}_2, \ldots\}$$

e.g. $F = x_1 \bar{x}_2 x_3$

sat assignments $\equiv$ any assignment s.t.
$x_1 = T,\ x_2 = F,\ x_3 = T$

random satisfying assignment to F:

Let $x_1 = T,\ x_2 = F,\ x_3 = T$
& pick $x_4 \ldots x_n$ randomly $\in \{T, F\}$

in general, satisfy literals in F
& pick other settings randomly

Two Conjunction Case:

Algorithm Attempt:

pick $i \in \{1,2\}$

set vars in conjunction i to "true"

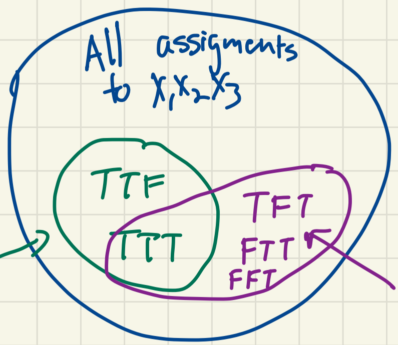
set other vars randomly

example: $x_1 x_2 \vee x_3$

pick 1

set $x_1 = x_2 = T$

set $x_3 = T$



Satisfy $x_1 x_2$

satisfy x_3

$\Pr[\text{output } TTF] = \frac{1}{2} \cdot \frac{1}{2} = 1/4$

Prob pick 1

Prob pick $x_3 = F$

$\Pr[\text{output } TFT] = \frac{1}{2} \cdot \frac{1}{4} = 1/8$

prob pick 2

prob pick $x_1 = T$, $x_2 = F$

$\Pr[\text{output } TTT] =$

$\frac{1}{2} \cdot \frac{1}{2} + \frac{1}{2} \cdot \frac{1}{4} = 3/8$

pick 1   pick $x_3 = T$ pick 2   pick $x_1 = T$, $x_2 = T$

not uniform:

1) 2nd conjunction has more sat assigments

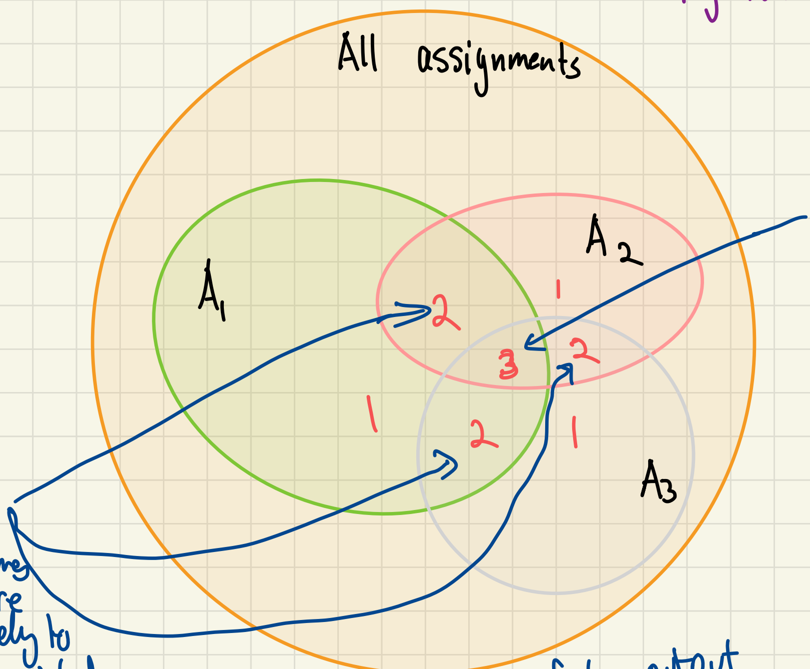
2) some assignments can be chosen multiple ways

main ideas to fix algorithm:

1) Choose conjunction proportionally to # sat assignments

2) if assignment can be output in >1 way, "correct" for it.

"rejection sampling"



3 times more likely to be picked. Correct by tossing coin of bias $\frac{1}{3}$ to decide if to output

2 times more likely to be picked. Correct by tossing bias $\frac{1}{2}$ coin to decide if to output

Let $A_i \leftarrow \{ \bar{x} = (x_1 \cdots x_n) \mid \bar{x}$ satisfies $C_i \}$

assignments that satisfy clause i

Algorithm: Input: $\phi = \bigvee_{i=1}^{m} C_i$ ← conjunctions

Let $A_i \leftarrow \{ \bar{x} = (x_1 \cdots x_n) \mid \bar{x} \text{ satisfies } C_i \}$

Repeat

Pick i with prob $\frac{|A_i|}{\sum_{j=1}^{m} |A_j|}$

we already saw → how to do this

Pick uniform assignment $\bar{b}$ in A_i

Let $t_{\bar{b}} \leftarrow |\{ j \mid \bar{b} \text{ satisfies } A_j \}|$ ← ≥ 1 since $\bar{b}$ satisfies A_i

Output $\bar{b}$ with prob $1/t_{\bar{b}}$

Until succeed

Uniformity:

$\forall \bar{b}$ s.t. $\bar{b}$ satisfies ϕ:

$$\Pr[\text{output } \bar{b} \text{ in round } i] = \frac{1}{t_{\bar{b}}} \sum_{\substack{j \in [m] \\ \text{s.t. } \bar{b} \in S_j}} \Pr[\text{pick } j \text{ in round } i] \cdot \frac{1}{|S_j|}$$

$$= \frac{1}{t_{\bar{b}}} \sum_{\substack{j \text{ s.t.} \\ \bar{b} \in S_j}} \frac{|S_j|}{\sum |S_k|} \cdot \frac{1}{|S_j|}$$

$$= \frac{1}{t_{\bar{b}}} \frac{t_{\bar{b}}}{\sum_k |S_k|} = \underbrace{\frac{1}{\sum_k |S_k|}}_{\text{same for all } \bar{b} \text{ that satisfy } \phi}$$

Runtime:

$$\Pr[\text{loop succeeds}] \geq \frac{1}{\max t_{\bar{b}}} \geq \frac{1}{m}$$

$$E[\#\text{ loops until succeeds}] \leq m$$

time per loop is poly$(m+n)$

Counting Problems

#P = class of problems that count # accept paths in poly-time nondeterministic Turing machines.

#P-complete:

- in #P
- every problem in #P has $\left.\begin{array}{l}\text{Turing reduction}\\\text{poly-time reduction}\end{array}\right\}$ to it

#SAT: # of assignments satisfying Boolean formula ϕ

#P-complete!

DNF is in P so should be easy!!!!

Is #DNF easier?

I hate to rain on your parade, but...

Not if $P \neq NP$

DeMorgan's law:

$$\overline{(A \vee B)} = \underbrace{\bar{A} \wedge \bar{B}}_{\text{conjunction}}$$

clause → conjunction

Why?

Given ϕ in CNF

ϕ is sat iff $\bar{\phi}$ has >1 unsat assignments

CNF (ϕ) DNF ($\bar{\phi}$)

$P = NP \Leftarrow$ ability to exactly count CNF in poly time $\Leftarrow$ ability to exactly count DNF in poly time

#DNF is #P-complete

Approximate Counting

Fully polynomial randomized approximation scheme (FPRAS)

Given ϕ, ε

s.t. $Z = \#$ sat assignments to ϕ

Output y s.t.

$$\frac{Z}{1+\varepsilon} \leq y \leq Z \cdot (1+\varepsilon)$$

with prob $\geq 3/4$

Hope: runtime poly in $|\phi|, \frac{1}{\varepsilon}$

pset 1 problem 1:

algorithm that satisfies "hope"

$\Rightarrow$ poly in $|\phi|, \frac{1}{\varepsilon}, \log 1/\delta$ $\Leftarrow$ prob of too much approx error "Confidence"

approx error

FPRAS for SAT?

FPRAS for SAT $\Rightarrow$ ptime algorithm for SAT:

Algorithm for SAT: Given formula ϕ

Call FPRAS on ϕ with $\varepsilon = 1/2$ ← any $\varepsilon > 0$ works

if output > 0 output "Satisfiable"

else output "unsatisfiable"

Correctness if ϕ satisfiable, $\#\phi \geq 1$ so

$$y > \frac{1}{1+\varepsilon} > 0 \Rightarrow \text{output "sat"} \checkmark$$

if ϕ unsatisfiable, $\#\phi = 0$ so

$$y = 0 \Rightarrow \text{output "unsat"} \checkmark$$

Exact vs. Approx Counting

Counting # SAT assignments to CNF is #P-complete

" " " " DNF " "

" " perfect matchings in graph " "

" " spanning trees in graph is in poly time

Is it hard to approx count?

CNF hard

DNF polytime ⟵ today

Matching polytime

Spanning trees polytime

your favorite problem?

Approx counting for DNF:

Fully polynomial randomized approximation scheme (FPRAS)

Given ϕ, ε

s.t. $z = \#$ sat assignments to ϕ

Output y s.t.

$$\frac{z}{1+\varepsilon} \le y \le z \cdot (1+\varepsilon)$$

with prob $\ge 3/4$

Will use:

(1) uniform generation of DNF sat assignments

(2) "Downward self-reducibility" of DNF

Downward self-reducibility: (dsr)

Can compute problem by solving smaller subproblems & putting together answers via poly time computation.

Why is #-DNF dsr.?

$$\#\phi(x_1 \cdots x_n) = \#\phi(x_1 = T, x_2, \ldots x_n) +$$

$$\#\phi(x_1 = F, x_2, \ldots x_n)$$

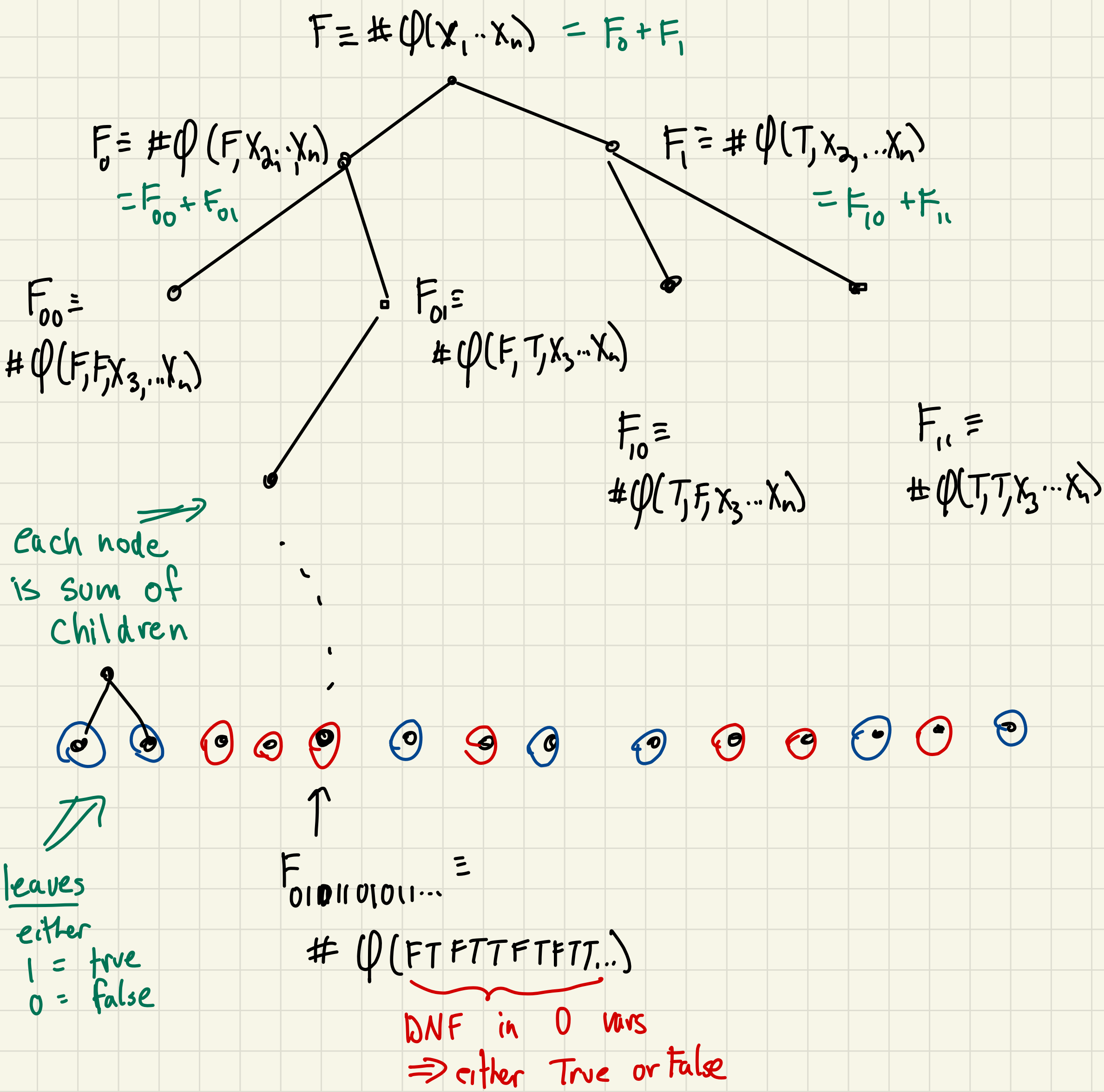
both are still DNFs but in n-1 vars.

e.g. $\#(x_1\bar{x}_2 \vee x_1x_2 \vee \bar{x}_2)$

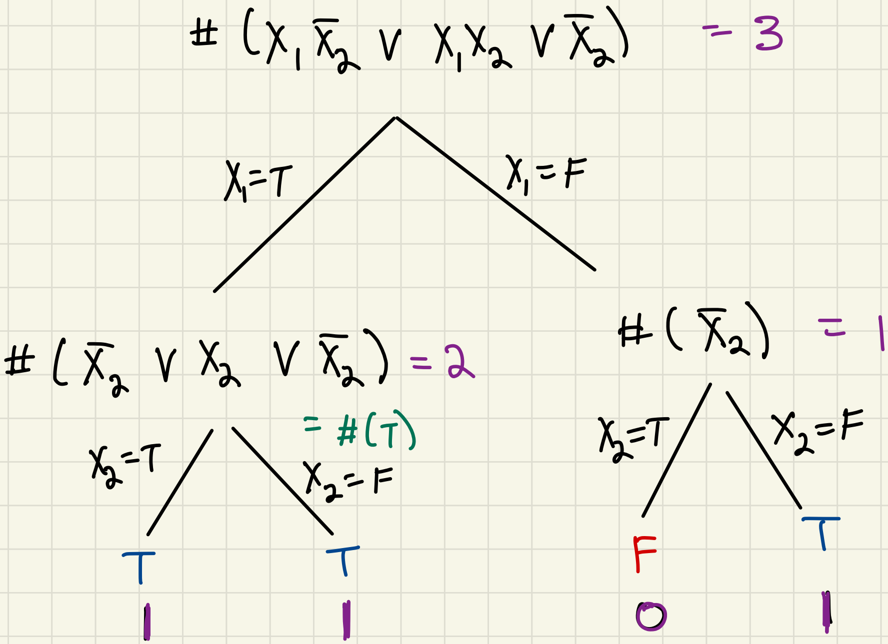
$= \#(\bar{x}_2 \vee x_2 \vee \bar{x}_2)$ ← # settings where $x_1 = T$

$+ \#(\bar{x}_2)$ ← # settings where $x_1 = F$

Downward Self-Reducibility Tree

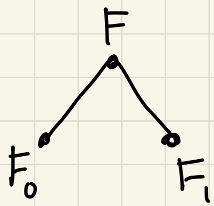


example



$\#(x_1\bar{x}_2 \vee x_1x_2 \vee \bar{x}_2) = 3$
$x_1=T$
$x_1=F$
$\#(\bar{x}_2 \vee x_2 \vee \bar{x}_2) = 2$
$= \#(T)$
$\#(\bar{x}_2) = 1$
$x_2=T$
$x_2=F$
$x_2=T$
$x_2=F$
T
T
F
T
1
1
0
1

Approximate Counting Algorithm for #DNF



Let $S_1 = \frac{F_1}{F}$ $\Rightarrow$ $F = \frac{F_1}{S_1}$

Fraction of SAT assignments s.t. $x_1 = T$

main insight: for DNF, we can estimate S_1 via sampling!

- uniformly generate k sat assignments ← we know how to do this for DNF!!
- $\tilde{S}_1 \leftarrow \frac{\#\text{ with } x_1 = T}{k}$

But how do we compute F_1?

recursively!

$$F_1 = \frac{F_{11}}{S_{11}}$$

F_{11} ← recurse

S_{11} ← estimate

So $F = \dfrac{F_{b_1}}{S_{b_1}} = \dfrac{F_{b_1 b_2}}{S_{b_1} \cdot S_{b_1 b_2}} = \dfrac{F_{b_1 b_2 b_3}}{S_{b_1} \cdot S_{b_1 b_2} \cdot S_{b_1 b_2 b_3}}$

$$\vdots$$

$$= \frac{1}{\prod_{i=1}^{n} S_{b_1 \cdots b_i}}$$

Potential Difficulties:

1. if $F_{b_1 \cdots b_n} = 0$ this doesn't work

2. Is approximation of $S_{b_1 \cdots b_i}$'s good enough? only get additive estimates :(

<u>Idea</u> Always take path of "larger" child

↑ might guess wrong when <u>both</u> have lots of SAT assignments but soon will show that is ok

<u>Claim</u> if always pick b_i s.t. $F_{b_1 \cdots b_i} > F_{b_1 \cdots \bar{b}_i}$ then always reach SAT assignment leaf. (so $F_{b_1 \cdots b_n} = 1$)

Idea: estimate each $S_{b_1 \dots b_i}$ to within $\frac{\varepsilon}{4n}$ additive error (using Chernoff bnds, need only $\text{poly}(\frac{2n}{\varepsilon}, \log 4n)$ samples to get error $< \frac{1}{4n}$)

$\Rightarrow$ if $1 \geq r \geq 1/2$

$$r + \frac{\varepsilon}{4n} \leq r\left(1 + \frac{\varepsilon}{4n \cdot r}\right) \leq r\left(1 + \frac{\varepsilon}{2n}\right)$$

$\uparrow$ union bnd over all i to get prob of error $< \frac{1}{4}$

* slight issue: might be estimating $1-r$ if pick wrong path. We will ignore this for now.

$$r - \frac{\varepsilon}{4n} \geq r\left(1 - \frac{\varepsilon}{4nr}\right) \geq r\left(1 - \frac{\varepsilon}{4n}\right)$$

Claim

$$\text{output} \leq \frac{F_{b_1}}{\tilde{S}_{b_1}} \leq \frac{F_{b_1 b_2}}{\tilde{S}_{b_1} \tilde{S}_{b_1 b_2}} \leq \dots \leq \frac{1}{\prod \tilde{S}_{b_1 \dots b_i}}$$

$$\leq \frac{\left(1 + \frac{\varepsilon}{4n}\right)^n}{\prod S_{b_1 \dots b_n}} = F \cdot \underbrace{\left(1 + \frac{\varepsilon}{4n}\right)^n}_{1 + \frac{\varepsilon}{4} + \frac{\left(\frac{\varepsilon}{4}\right)^2}{2!} + \dots \,\leq\, 1 + \frac{\varepsilon}{2}} \leq F\left(1 + \frac{\varepsilon}{2}\right)$$

Taylor series

similarly, $\text{output} \geq \frac{F}{1+\varepsilon}$

$\blacksquare$

Recursive Algorithm

- estimate S_0, S_1 from unif generated SAT assignments
- let $b_1 \leftarrow \arg\max \{S_0, S_1\}$
- recurse on F_{b_1}

runtime?

$\leq \underbrace{n}_{\text{\#recursions}} \cdot \underbrace{\text{\#samples to get } \frac{\varepsilon}{4n} \text{ additive error}}_{\text{poly}\left(\frac{\varepsilon}{4n}\right)^{-1} \cdot \left(\frac{1}{4n}\right)^{-1}} \cdot \underbrace{\text{runtime of uniform generator}}_{\text{poly in } n}$

$\left(\frac{\varepsilon}{4n}\right)^{-1}$: approx error

$\left(\frac{1}{4n}\right)^{-1}$: confidence error

total: $\text{poly}\left(n, \frac{1}{\varepsilon}\right)$

$$\Pr[\text{algorithm fails}] \leq \sum_{\substack{\text{recursion level} \\ i=1}}^{n} \Pr[\text{estimate bad}] \leq n \cdot \frac{1}{4n} \leq \frac{1}{4}.$$

Works for any d.s.r. problem!

poly time (almost)-uniform-generation of solutions

$\Downarrow$ $\Uparrow$ what about this direction?

poly time approximate counting of # solns

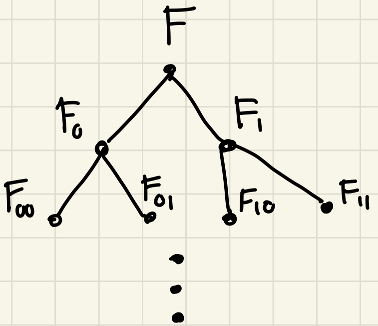
<u>Thm</u> [Jerrum Valiant Vazirani] for any problem in NP that is d.s.r.:

ptime approx counting of # solutions $\Longleftrightarrow$ ptime almost uniform generation

(easier case)

(Perfect) counting for # DNF $\Rightarrow$

(perfect) Uniform generation



Recursive algorithm:

at $b_1 \dots b_i$,

use (perfect) counter to compute

$$r_0 = F_{b_1 \dots b_i 0}$$

$$r_1 = F_{b_1 \dots b_i 1}$$

go left with prob $\frac{r_0}{r_0 + r_1}$

& right o.w.

Claim (1) always reach SAT assignment

since never take branch with 0 SAT assignments underneath

(2) $Pr[\text{output } \underbrace{b_1 \dots b_n}_{\text{SAT assignment}}] = \frac{F_{b_1}}{F} \cdot \frac{F_{b_1 b_2}}{F_{b_1}} \cdot \frac{F_{b_1 b_2 b_3}}{F_{b_1 b_2}} \cdot \dots \cdot \frac{1}{F_{b_1 b_2 \dots b_n}}$

$= \frac{1}{F}$ ← same for every sat assignment

Question what if only have approx counter?

Answer $\text{RHS} \le \frac{1}{F}\left(\frac{1+\varepsilon'}{1-\varepsilon'}\right)^n \le \frac{1}{F}\cdot\frac{1}{1-\varepsilon}$

if choose $\varepsilon' < \frac{\varepsilon}{2n}$

$\Longrightarrow$ close to uniform generation of sat assignments