

Randomized Algorithms

The Monte Carlo Method

Joseph Chuang-Chieh Lin

Department of Computer Science & Engineering,
National Taiwan Ocean University

Spring 2026



Outline

- 1 Introduction
- 2 The DNF counting problem
- 3 DNF counting algorithms
- 4 Counting independent sets in a graph.
 - Markov Chain Monte Carlo (MCMC) Algorithms



Outline

- 1 Introduction
- 2 The DNF counting problem
- 3 DNF counting algorithms
- 4 Counting independent sets in a graph.
 - Markov Chain Monte Carlo (MCMC) Algorithms



Introduction

- The Monte Carlo method: estimating values through sampling and simulation.
- Widely used in almost all areas of physical sciences and engineering.



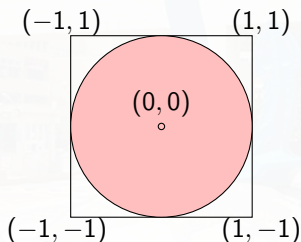
Introduction (example)

- As a motivating example, consider the following approach for estimating the constant π .



Estimating π

- Let (X, Y) be a point chosen uniformly at random in a 2×2 square centered at the origin $(0, 0)$.



- [demo code on Colab](#)



Estimating π (indicator variable)

- Define the random variable

$$Z = \begin{cases} 1, & \text{if } \sqrt{X^2 + Y^2} \leq 1, \\ 0, & \text{otherwise.} \end{cases}$$

- The probability that $Z = 1$ is the ratio

$$\Pr[Z = 1] = \frac{\text{area of unit circle}}{\text{area of square}} = \frac{\pi}{4}.$$



Estimating π (sampling)

- Run this experiment m times, and let Z_i be the value of Z in the i -th run.
- Define

$$W = \sum_{i=1}^m Z_i.$$

- Then

$$\mathbb{E}[W] = \sum_{i=1}^m \mathbb{E}[Z_i] = m \cdot \frac{\pi}{4}.$$

- Hence

$$W' = \frac{4}{m} W$$

is a natural estimate for π .



Estimating π (Chernoff bound)

- Using Chernoff bounds on W , we can bound the error of W' .
- For $0 < \varepsilon < 1$,

$$\begin{aligned}\Pr[|W' - \pi| \geq \varepsilon\pi] &= \Pr[|W - m\pi/4| \geq \varepsilon m\pi/4] \\ &= \Pr[|W - \mathbb{E}[W]| \geq \varepsilon\mathbb{E}[W]] \\ &\leq 2e^{-m\pi\varepsilon^2/12}.\end{aligned}$$

- Thus, with sufficiently many samples, we can obtain an arbitrarily tight approximation of π with high probability.



(ϵ, Δ) -approximation randomized algorithm

Definition.

- A randomized algorithm gives an (ϵ, Δ) -approximation for a value V if its output X satisfies

$$\Pr[|X - V| \leq \epsilon V] \geq 1 - \Delta.$$

- Our method for estimating π gives an (ϵ, Δ) -approximation as long as $\epsilon < 1$ and m is large enough.
- From the Chernoff bound:

$$2e^{-m\pi\epsilon^2/12} \leq \Delta \implies m \geq \frac{12}{\pi\epsilon^2} \ln \frac{2}{\Delta}.$$

- Further reading: [Probably Approximately Correct \(PAC\)](#)



A general sampling theorem

Theorem 1.

Let X_1, \dots, X_m be independent and identically distributed indicator random variables with $\mu = \mathbb{E}[X_i]$. If

$$m \geq \frac{3 \ln(2/\Delta)}{\varepsilon^2 \mu},$$

then

$$\Pr \left[\left| \frac{1}{m} \sum_{i=1}^m X_i - \mu \right| \geq \varepsilon \mu \right] \leq \Delta.$$

Thus, m samples provide an (ε, Δ) -approximation for μ .



Approximation schemes

- For some problems, an efficient (polynomial-time) exact algorithm would imply $P = NP$.



Approximation schemes

- For some problems, an efficient (polynomial-time) exact algorithm would imply $P = NP$.
- It is therefore unlikely that such exact algorithms exist.



Approximation schemes

- For some problems, an efficient (polynomial-time) exact algorithm would imply $P = NP$.
- It is therefore unlikely that such exact algorithms exist.
- Instead, we study approximation algorithms and approximation schemes:
 - Polynomial-time approximation schemes (PTAS)
 - Fully polynomial-time approximation schemes (FPTAS)
 - Polynomial randomized approximation schemes (PRAS)
 - Fully polynomial randomized approximation schemes (FPRAS)



Approximation schemes

- For some problems, an efficient (polynomial-time) exact algorithm would imply $P = NP$.
- It is therefore unlikely that such exact algorithms exist.
- Instead, we study approximation algorithms and approximation schemes:
 - Polynomial-time approximation schemes (PTAS)
 - Fully polynomial-time approximation schemes (FPTAS)
 - Polynomial randomized approximation schemes (PRAS)
 - Fully polynomial randomized approximation schemes (FPRAS)
- In this lecture, we will focus on PRAS and FPRAS.



Counting problems

- We focus on counting problems that map inputs x to values $V(x)$.
- Example:
 - Given a graph, we may want an approximation to the *number* of independent sets in the graph.
 - Given a Boolean formula in Disjunctive Normal Form (DNF), which is a disjunction of conjunctions of literals, count the *number* of satisfying assignments.



PRAS

Definition (PRAS)

- A *polynomial randomized approximation scheme* (PRAS) for a problem is a randomized algorithm such that, given input x and parameters ε, Δ with $0 < \varepsilon, \Delta < 1$, it outputs an (ε, Δ) -approximation to $V(x)$ in time $\text{poly}(|x|)$.

FPRAS

Definition (FPRAS)

- A *fully polynomial randomized approximation scheme* (FPRAS) for a problem is a randomized algorithm such that, given input x and parameters ε, Δ with $0 < \varepsilon, \Delta < 1$, it outputs an (ε, Δ) -approximation to $V(x)$ in time

$$\text{poly}(|x|, 1/\varepsilon, \ln(1/\Delta)).$$



Outline

- 1 Introduction
- 2 The DNF counting problem
- 3 DNF counting algorithms
- 4 Counting independent sets in a graph.
 - Markov Chain Monte Carlo (MCMC) Algorithms



The DNF counting problem

- We consider the problem of counting the number of satisfying assignments of a Boolean formula in *disjunctive normal form* (DNF).

Definition

- A DNF formula is a disjunction of clauses

$$F = C_1 \vee C_2 \vee \cdots \vee C_t, \quad t \in \mathbb{N}$$

where each clause is a conjunction of literals.

- Example:

$$(X_1 \wedge \neg X_2 \wedge X_3) \vee (X_2 \wedge X_4) \vee (X_1 \wedge X_3 \wedge \neg X_4).$$



#P and #P-completeness

Class #P

A problem is in the class #P if there is a polynomial-time nondeterministic Turing machine (algorithm) such that, for any input I , the number of accepting computations equals the number of solutions associated with I .

- Clearly, a #P problem is at least as hard as the corresponding NP decision problem.

#P-complete

A problem is #P-complete if it is in #P and every problem in #P can be reduced to it in polynomial time.

- Examples:
 - Counting Hamiltonian cycles in a graph.
 - Counting perfect matchings in a bipartite graph.



Example: From CNF to DNF via Negation

- CNF formula H with three clauses:

$$H = (x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_1 \vee x_2) \wedge (\neg x_2 \vee \neg x_3).$$

- Negate H and apply De Morgan's laws:

$$\begin{aligned}\neg H &= \neg[(x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_1 \vee x_2) \wedge (\neg x_2 \vee \neg x_3)] \\ &= \neg(x_1 \vee \neg x_2 \vee x_3) \vee \neg(\neg x_1 \vee x_2) \vee \neg(\neg x_2 \vee \neg x_3) \\ &= (\neg x_1 \wedge x_2 \wedge \neg x_3) \vee (x_1 \wedge \neg x_2) \vee (x_2 \wedge x_3).\end{aligned}$$

- $\neg H$ is in DNF (a disjunction of conjunctions of literals). Each CNF clause of H becomes one conjunction in $\neg H$.



Hardness of DNF counting

- Counting the number of satisfying assignments of a DNF formula is #P-complete.
- Sketch:
 - Given any CNF formula H , apply De Morgan's laws to obtain a DNF formula \bar{H} (the negation of H) with the same number of variables and clauses.



Hardness of DNF counting

- Counting the number of satisfying assignments of a DNF formula is #P-complete.
- Sketch:
 - Given any CNF formula H , apply De Morgan's laws to obtain a DNF formula \bar{H} (the negation of H) with the same number of variables and clauses.
 - If H has n variables, then

$$H \text{ is satisfiable} \iff \#\{\text{satisfying assignments of } \bar{H}\} < 2^n.$$



Hardness of DNF counting

- Counting the number of satisfying assignments of a DNF formula is #P-complete.
- Sketch:
 - Given any CNF formula H , apply De Morgan's laws to obtain a DNF formula \bar{H} (the negation of H) with the same number of variables and clauses.
 - If H has n variables, then

$$H \text{ is satisfiable} \iff \#\{\text{satisfying assignments of } \bar{H}\} < 2^n.$$

- Hence DNF counting is at least as hard as SAT (NP-hard).



Hardness of DNF counting

- Counting the number of satisfying assignments of a DNF formula is #P-complete.
- Sketch:
 - Given any CNF formula H , apply De Morgan's laws to obtain a DNF formula \bar{H} (the negation of H) with the same number of variables and clauses.
 - If H has n variables, then

$$H \text{ is satisfiable} \iff \#\{\text{satisfying assignments of } \bar{H}\} < 2^n.$$

- Hence DNF counting is at least as hard as SAT (NP-hard).
 - #satisfying assignment of $H = 2^n - \#\text{satisfying assignment of } \bar{H}$.
 - Hence we can obtain a polynomial time reduction from #SAT to DNF-counting, where #SAT is known to be #P-complete.



Why approximation for DNF counting?

- It is unlikely there is a polynomial-time algorithm that computes the exact number of solutions of a $\#P$ -complete problem.
 - Such an algorithm would imply $P = NP$.
- Therefore, it is interesting to find an approximation scheme, such as an FPRAS, for counting satisfying assignments of a DNF formula.



Outline

- 1 Introduction
- 2 The DNF counting problem
- 3 DNF counting algorithms**
- 4 Counting independent sets in a graph.
 - Markov Chain Monte Carlo (MCMC) Algorithms



A naïve sampling algorithm

- $c(F)$: the number of satisfying assignments of a DNF formula F .
- We assume $c(F) > 0$ (it is easy to check whether $c(F) = 0$ first).

DNF counting algorithm I

- **Input:** DNF formula F with n variables.
 - **Output:** Y , as an approximation of $c(F)$.
- 1 $X \leftarrow 0$.
 - 2 For $k \leftarrow 1$ to m :
 - 1 Generate a random assignment for the n variables, chosen uniformly at random from all 2^n assignments.
 - 2 If the random assignment satisfies F , set $X \leftarrow X + 1$.
 - 3 Return $Y \leftarrow (X/m) 2^n$.

Analysis of the naïve algorithm

- Let

$$X_k = \begin{cases} 1, & \text{if the } k\text{-th assignment satisfies } F, \\ 0, & \text{otherwise.} \end{cases}$$

- Then $\Pr[X_k = 1] = c(F)/2^n$.
- Let $X = \sum_{k=1}^m X_k$, so

$$\mathbb{E}[X] = m \cdot \frac{c(F)}{2^n}.$$

- Hence

$$\mathbb{E}[Y] = \frac{\mathbb{E}[X] \cdot 2^n}{m} = c(F),$$

i.e., Y is an unbiased estimator of $c(F)$.



Limitations of the naïve algorithm

Using Theorem 1 with $\mu = c(F)/2^n$:

- We need

$$m \geq \frac{3 \cdot 2^n \ln(2/\Delta)}{\varepsilon^2 c(F)}.$$

- If $c(F) \geq 2^n/\alpha(n)$ for some polynomial α , then m is polynomial in n , $1/\varepsilon$ and $\ln(1/\Delta)$.
- But if $c(F) = \text{poly}(n)$, then

$$m = \Theta\left(\frac{2^n}{c(F)}\right),$$

which is not necessarily polynomial.

Key issue

The set of satisfying assignments may be **too sparse** in the space of all 2^n assignments.



Revising the algorithm

- We revise the naïve algorithm to obtain an FPRAS.
- Let

$$F = C_1 \vee C_2 \vee \cdots \vee C_t$$

be a DNF formula, with clauses C_i .

- Assume w.l.o.g. that no clause contains both a variable and its negation.
- If clause C_i has ℓ_i literals, then there are exactly $2^{n-\ell_i}$ satisfying assignments for C_i .



Sets S_{C_i} , U and S

- Let S_{C_i} be the set of assignments that satisfy clause C_i .
- Define a set for “over-estimate”:

$$U = \{(i, a) : 1 \leq i \leq t, a \in S_{C_i}\}.$$

- Then

$$|U| = \sum_{i=1}^t |S_{C_i}|.$$

- We want to estimate

$$c(F) = \left| \bigcup_{i=1}^t S_{C_i} \right|.$$



Sets S_{C_i} , U and S

- Let S_{C_i} be the set of assignments that satisfy clause C_i .
- Define a set for “over-estimate”:

$$U = \{(i, a) : 1 \leq i \leq t, a \in S_{C_i}\}.$$

- Then

$$|U| = \sum_{i=1}^t |S_{C_i}|.$$

- We want to estimate

$$c(F) = \left| \bigcup_{i=1}^t S_{C_i} \right|.$$

Clearly $c(F) \leq |U|$.

(\because an assignment satisfying multiple clauses appears multiple times in U .)



Defining a dense subset S

- Construct a subset $S \subseteq U$ that has exactly one pair for each satisfying assignment of F .
- For each satisfying assignment a of F , choose the smallest index i such that $a \in S_{C_i}$ and include (i, a) in S .
- Formally,

$$S = \{(i, a) : 1 \leq i \leq t, a \in S_{C_i}, a \notin S_{C_j} \text{ for } j < i\}.$$

- Then $|S| = c(F)$.



Defining a dense subset S

- Construct a subset $S \subseteq U$ that has exactly one pair for each satisfying assignment of F .
- For each satisfying assignment a of F , choose the smallest index i such that $a \in S_{C_i}$ and include (i, a) in S .
- Formally,

$$S = \{(i, a) : 1 \leq i \leq t, a \in S_{C_i}, a \notin S_{C_j} \text{ for } j < i\}.$$

- Then $|S| = c(F)$.
- Moreover, each assignment can satisfy at most t clauses, so

$$\frac{|S|}{|U|} \geq \frac{1}{t},$$

i.e., S is relatively dense in U .



An Illustrating Example (1/4)

- Consider three variables x_1, x_2, x_3 and the DNF formula

$$F = C_1 \vee C_2 \vee C_3,$$

where

$$C_1 = x_1 \wedge x_2, \quad C_2 = x_2 \wedge x_3, \quad C_3 = x_1 \wedge x_3.$$

- An assignment $a = (x_1, x_2, x_3) \in \{0, 1\}^3$ satisfies C_i iff:

$$S_{C_1} = \{110, 111\}, \quad S_{C_2} = \{011, 111\}, \quad S_{C_3} = \{101, 111\}.$$

- Satisfying assignments of F :

$$\bigcup_{i=1}^3 S_{C_i} = \{110, 111, 011, 101\}.$$

- Hence the number of satisfying assignments is $c(F) = 4$.



An Illustrating Example (2/4): the over-estimate set U

- Recall the definition

$$U = \{(i, a) : 1 \leq i \leq t, a \in S_G\}.$$

- In our example, $t = 3$, so

$$U = \{(1, 110), (1, 111)\} \cup \{(2, 011), (2, 111)\} \cup \{(3, 101), (3, 111)\}.$$

- Thus $|U| = 6$.
- Note:** the assignment 111 appears three times in U , once for each clause it satisfies.



An Illustrating Example (3/4): Constructing $S \subseteq U$

- For each satisfying assignment a of F , choose the *smallest* index i such that $a \in S_{C_i}$, and include (i, a) in S .
- For our four satisfying assignments:

a	clauses satisfied	chosen pair
110	C_1	$(1, 110)$
011	C_2	$(2, 011)$
101	C_3	$(3, 101)$
111	C_1, C_2, C_3	$(1, 111)$

- Hence

$$S = \{(1, 110), (2, 011), (3, 101), (1, 111)\}.$$

- Every satisfying assignment appears *exactly once* in S , so

$$|S| = 4 = c(F).$$



An Illustrating Example (4/4): Density of S inside U

- In general, each assignment can satisfy at most t clauses, so each assignment appears in at most t pairs in U .
- In our example, $t = 3$, and we found

$$|S| = 4, \quad |U| = 6.$$

- Thus

$$\frac{|S|}{|U|} = \frac{4}{6} = \frac{2}{3} \geq \frac{1}{3} = \frac{1}{t}.$$

- This illustrates the general property:

$$\frac{|S|}{|U|} \geq \frac{1}{t},$$

i.e., S is relatively dense in U .



DNF counting algorithm II

DNF counting algorithm II (constructing S)

- **Input:** DNF formula F with n variables and clauses C_1, \dots, C_t .
 - **Output:** Y , an approximation of $c(F)$.
- 1 $X \leftarrow 0$.
 - 2 For $k = 1$ to m :
 - 1 Choose $i \in \{1, 2, \dots, t\}$ with probability $|S_{C_i}| / \sum_{j=1}^t |S_{C_j}|$, and then choose an assignment $a \in S_{C_i}$ uniformly at random.
 - 2 If a does not satisfy any C_j with $j < i$, then set $X \leftarrow X + 1$.
 - 3 Return

$$Y \leftarrow \frac{X}{m} \sum_{i=1}^t |S_{C_i}|.$$



Analysis of algorithm II

- For each pair (i, a) with $a \in S_{C_i}$:

$$\begin{aligned}\Pr[(i, a) \text{ is chosen}] &= \Pr[i \text{ is chosen}] \cdot \Pr[a \text{ is chosen} \mid i] \\ &= \frac{|S_{C_i}|}{|U|} \cdot \frac{1}{|S_{C_i}|} = \frac{1}{|U|}.\end{aligned}$$

- Thus, we choose a uniformly random element of U .



Analysis of algorithm II

- For each pair (i, a) with $a \in S_{C_i}$:

$$\begin{aligned}\Pr[(i, a) \text{ is chosen}] &= \Pr[i \text{ is chosen}] \cdot \Pr[a \text{ is chosen} \mid i] \\ &= \frac{|S_{C_i}|}{|U|} \cdot \frac{1}{|S_{C_i}|} = \frac{1}{|U|}.\end{aligned}$$

- Thus, we choose a uniformly random element of U .
- The probability that the chosen pair lies in S is $\frac{|S|}{|U|} = \frac{c(F)}{|U|} \geq \frac{1}{t}$.



Analysis of algorithm II

- For each pair (i, a) with $a \in S_{C_i}$:

$$\begin{aligned}\Pr[(i, a) \text{ is chosen}] &= \Pr[i \text{ is chosen}] \cdot \Pr[a \text{ is chosen} \mid i] \\ &= \frac{|S_{C_i}|}{|U|} \cdot \frac{1}{|S_{C_i}|} = \frac{1}{|U|}.\end{aligned}$$

- Thus, we choose a uniformly random element of U .
- The probability that the chosen pair lies in S is $\frac{|S|}{|U|} = \frac{c(F)}{|U|} \geq \frac{1}{t}$.
- So the indicator variables of “chosen pair is in S ” have mean $\geq 1/t$.



Analysis of algorithm II

- For each pair (i, a) with $a \in S_{C_i}$:

$$\begin{aligned}\Pr[(i, a) \text{ is chosen}] &= \Pr[i \text{ is chosen}] \cdot \Pr[a \text{ is chosen} \mid i] \\ &= \frac{|S_{C_i}|}{|U|} \cdot \frac{1}{|S_{C_i}|} = \frac{1}{|U|}.\end{aligned}$$

- Thus, we choose a uniformly random element of U .
- The probability that the chosen pair lies in S is $\frac{|S|}{|U|} = \frac{c(F)}{|U|} \geq \frac{1}{t}$.
- So the indicator variables of “chosen pair is in S ” have mean $\geq 1/t$.
- By Theorem 1, with

$$m = \left\lceil \frac{3t}{\varepsilon^2} \ln\left(\frac{2}{\Delta}\right) \right\rceil$$

we obtain an (ε, Δ) -approximation of $c(F)$.



Theorem 2

Theorem 2

- DNF counting algorithm II is an FPRAS for the DNF counting problem when

$$m = \left\lceil \frac{3t}{\varepsilon^2} \ln\left(\frac{2}{\Delta}\right) \right\rceil.$$

- The running time is polynomial in t , $1/\varepsilon$, and $\ln(1/\Delta)$.



Outline

- 1 Introduction
- 2 The DNF counting problem
- 3 DNF counting algorithms
- 4 Counting independent sets in a graph.**
 - Markov Chain Monte Carlo (MCMC) Algorithms



Approximate uniform sampling

- We next present a *general reduction*:
 - If we can **sample almost uniformly from the solution space of a self-reducible combinatorial problem**, then we can approximately count the number of solutions.

Self-Reducible Combinatorial Problem

A problem where solutions for a large instance can be systematically reconstructed from solutions of polynomially many smaller instances of the same problem.

- We illustrate this technique for the problem of **counting independent sets in a graph**.



Independent Set

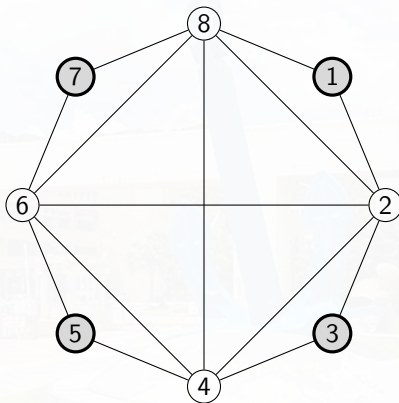


Fig.: $\{2, 4, 6, 8\}$ forms a clique and $\{1, 3, 5, 7\}$ is an independent set.



ϵ -uniform sample and FPAUS

Let Ω be a finite sample space and w be the (random) output of a sampling algorithm.

ϵ -uniform sample

The algorithm generates an ϵ -uniform sample of Ω if for any subset $S \subseteq \Omega$,

$$|\Pr[w \in S] - |S|/|\Omega|| \leq \epsilon.$$

FPAUS

A sampling algorithm is a *fully polynomial almost uniform sampler* (FPAUS) for a problem if, given input x and parameter $\epsilon > 0$, it generates an ϵ -uniform sample of $\Omega(x)$ and runs in time $\text{poly}(|x|, \ln(1/\epsilon))$.



From FPAUS to FPRAS (independent sets)

- Consider an FPAUS for independent sets.
 - Input: a graph $G = (V, E)$ and parameter ε .
 - Sample space $\Omega(G)$: all independent sets in G .
 - We want an FPRAS for $|\Omega(G)|$, the number of independent sets.
 - Let $m = |E|$ and fix an ordering e_1, e_2, \dots, e_m of the edges.
 - Let E_i be the set of the first i edges, $G_i = (V, E_i)$.
 - Let $\Omega(G_i)$ be the set of independent sets in G_i .
- ★ **Note:** G_0 has no edges, so $|\Omega(G_0)| = 2^{|V|}$.



Decomposition of $|\Omega(G)|$

- Define ratios

$$r_i = \frac{|\Omega(G_i)|}{|\Omega(G_{i-1})|}, \quad i = 1, 2, \dots, m.$$

- Then

$$|\Omega(G)| = |\Omega(G_m)| = |\Omega(G_0)| \cdot \frac{|\Omega(G_1)|}{|\Omega(G_0)|} \cdot \frac{|\Omega(G_2)|}{|\Omega(G_1)|} \cdots \frac{|\Omega(G_m)|}{|\Omega(G_{m-1})|} = 2^n \prod_{i=1}^m r_i,$$

where $n = |V|$.

Decomposition of $|\Omega(G)|$

- Define ratios

$$r_i = \frac{|\Omega(G_i)|}{|\Omega(G_{i-1})|}, \quad i = 1, 2, \dots, m.$$

- Then

$$|\Omega(G)| = |\Omega(G_m)| = |\Omega(G_0)| \cdot \frac{|\Omega(G_1)|}{|\Omega(G_0)|} \cdot \frac{|\Omega(G_2)|}{|\Omega(G_1)|} \cdots \frac{|\Omega(G_m)|}{|\Omega(G_{m-1})|} = 2^n \prod_{i=1}^m r_i,$$

where $n = |V|$.

- If we obtain good estimates \hat{r}_i for each r_i , then

$$|\widehat{\Omega(G)}| = 2^n \prod_{i=1}^m \hat{r}_i$$

will be an approximation to $|\Omega(G)|$.



Lemma 1 (error control)

Lemma 1

Suppose that for all i , $1 \leq i \leq m$, the estimate \hat{r}_i is an $(\varepsilon/(2m), \Delta/m)$ -approximation for r_i . Let $R = \prod_{i=1}^m \frac{\hat{r}_i}{r_i}$. Then

$$\Pr[|R - 1| \leq \varepsilon] \geq 1 - \Delta.$$

Idea of proof:

- Apply the definition of (ε, Δ) -approximation to each \hat{r}_i .
- Use the union bound over all i .
- Bound the product $\prod_i (\hat{r}_i / r_i)$ using $(1 \pm x)$ estimates.



Proof of Lemma 1 (1/2)

- Let E_i be the event

$$E_i = \left\{ |\tilde{r}_i - r_i| \leq \frac{\varepsilon}{2m} r_i \right\}.$$

Then by assumption $\Pr[E_i] \geq 1 - \Delta/m$ for each i .



Proof of Lemma 1 (1/2)

- Let E_i be the event

$$E_i = \left\{ |\tilde{r}_i - r_i| \leq \frac{\varepsilon}{2m} r_i \right\}.$$

Then by assumption $\Pr[E_i] \geq 1 - \Delta/m$ for each i .

- Step 1: all estimates good with high probability.** Let $E = \bigcap_{i=1}^m E_i$. By the union bound,

$$\Pr[E] = 1 - \Pr[\neg E] \geq 1 - \sum_{i=1}^m \Pr[\neg E_i] \geq 1 - m \cdot \frac{\Delta}{m} = 1 - \Delta.$$



Proof of Lemma 1 (2/2)

- **Step 2: bound the product when all E_i hold.** On E_i we have

$$1 - \frac{\varepsilon}{2m} \leq \frac{\tilde{r}_i}{r_i} \leq 1 + \frac{\varepsilon}{2m}.$$

- Hence on E (assuming $0 < \varepsilon \leq 1$),

$$\left(1 - \frac{\varepsilon}{2m}\right)^m \leq R \leq \left(1 + \frac{\varepsilon}{2m}\right)^m.$$

- Thus on E , $|R - 1| \leq \varepsilon$. Since $\Pr[E] \geq 1 - \Delta$, we conclude

$$\Pr[|R - 1| \leq \varepsilon] \geq 1 - \Delta.$$



Proof of Lemma 1 (2/2)

- **Step 2: bound the product when all E_i hold.** On E_i we have

$$1 - \frac{\varepsilon}{2m} \leq \frac{\tilde{r}_i}{r_i} \leq 1 + \frac{\varepsilon}{2m}.$$

- Hence on E (assuming $0 < \varepsilon \leq 1$),

$$\left(1 - \frac{\varepsilon}{2m}\right)^m \leq R \leq \left(1 + \frac{\varepsilon}{2m}\right)^m.$$

- Thus on E , $|R - 1| \leq \varepsilon$. Since $\Pr[E] \geq 1 - \Delta$, we conclude

$$\Pr[|R - 1| \leq \varepsilon] \geq 1 - \Delta.$$



Proof of Lemma 1 (2/2)

- **Step 2: bound the product when all E_i hold.** On E_i we have

$$1 - \frac{\varepsilon}{2m} \leq \frac{\tilde{r}_i}{r_i} \leq 1 + \frac{\varepsilon}{2m}.$$

- Hence on E (assuming $0 < \varepsilon \leq 1$),

$$1 - \varepsilon \leq (1 - \varepsilon/2) \leq \left(1 - \frac{\varepsilon}{2m}\right)^m \leq R \leq \left(1 + \frac{\varepsilon}{2m}\right)^m \leq (1 + \varepsilon/2) \leq 1 + \varepsilon.$$

- Thus on E , $|R - 1| \leq \varepsilon$. Since $\Pr[E] \geq 1 - \Delta$, we conclude

$$\Pr[|R - 1| \leq \varepsilon] \geq 1 - \Delta.$$



Estimating r_i

Algorithm to estimate r_i

- **Input:** graphs $G_{i-1} = (V, E_{i-1})$ and $G_i = (V, E_i)$.
 - **Output:** estimate \hat{r}_i of r_i .
- 1 $X \leftarrow 0$.
 - 2 Repeat for $M = \left\lceil 1296 m^2 \varepsilon^{-2} \ln\left(\frac{2m}{\Delta}\right) \right\rceil$ independent trials:
 - a Generate an $(\varepsilon/(6m))$ -uniform sample from $\Omega(G_{i-1})$.
 - b If the sample is an independent set in G_i , set $X \leftarrow X + 1$.
 - 3 Return $\hat{r}_i \leftarrow X/M$.



Lemma 2 (quality of \hat{r}_i)

Lemma 2

When $m \geq 1$ and $0 < \varepsilon \leq 1$, the above procedure for estimating r_i yields an $(\varepsilon/(2m), \Delta/m)$ -approximation for r_i .

Sketch of proof:

- Show that $r_i = |\Omega(G_i)|/|\Omega(G_{i-1})| \geq 1/2$: adding a single edge removes at most half of the independent sets.
- Use the ε -uniformity of the sampler to bound the bias of \hat{r}_i :

$$|\mathbb{E}[\hat{r}_i] - r_i| \leq \frac{\varepsilon}{6m}.$$

- Apply Theorem 1 with appropriate parameters and combine bounds to obtain the $(\varepsilon/(2m), \Delta/m)$ guarantee.



Lemma 2 – Proof (1)

Step 1: $r_i \geq 1/2$.

- G_i is obtained from G_{i-1} by adding a single edge $e_i = (u, v)$.
- **Note:** An independent set in G_i is also an independent set in G_{i-1} .
- Each independent set in $\Omega(G_{i-1}) \setminus \Omega(G_i)$ contains both u and v .



Lemma 2 – Proof (1)

Step 1: $r_i \geq 1/2$.

- G_i is obtained from G_{i-1} by adding a single edge $e_i = (u, v)$.
- **Note:** An independent set in G_i is also an independent set in G_{i-1} .
- Each independent set in $\Omega(G_{i-1}) \setminus \Omega(G_i)$ contains both u and v .
 - Independent sets disappear due to the newly added edge (u, v) .



Lemma 2 – Proof (1) (contd.)

- Associate each $I \in \Omega(G_{i-1}) \setminus \Omega(G_i)$ with an independent set $I \setminus \{v\} \in \Omega(G_i)$.
- In this mapping, note that $I' \in \Omega(G_i)$ is associated with no more than one independent set $I' \cup \{v\} \in \Omega(G_{i-1}) \setminus \Omega(G_i)$, thus $|\Omega(G_{i-1}) \setminus \Omega(G_i)| \leq |\Omega(G_i)|$.



Lemma 2 – Proof (1) (contd.)

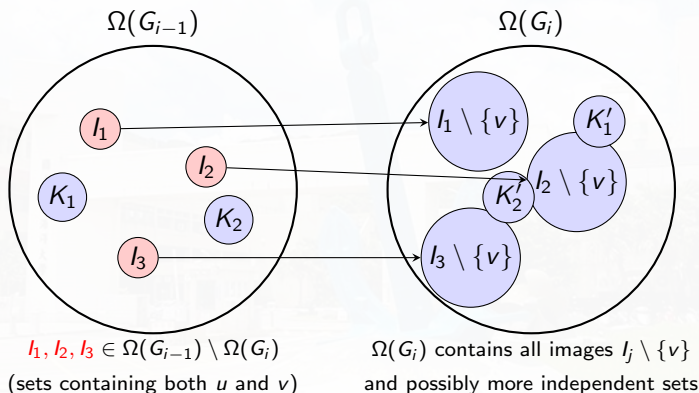
- Associate each $I \in \Omega(G_{i-1}) \setminus \Omega(G_i)$ with an independent set $I \setminus \{v\} \in \Omega(G_i)$.
- In this mapping, note that $I' \in \Omega(G_i)$ is associated with no more than one independent set $I' \cup \{v\} \in \Omega(G_{i-1}) \setminus \Omega(G_i)$, thus $|\Omega(G_{i-1}) \setminus \Omega(G_i)| \leq |\Omega(G_i)|$.
- It follows that

$$r_i = \frac{|\Omega(G_i)|}{|\Omega(G_{i-1})|} = \frac{|\Omega(G_i)|}{|\Omega(G_i)| + |\Omega(G_{i-1}) \setminus \Omega(G_i)|} \geq \frac{1}{2}.$$

- Hence at most half of the independent sets of G_{i-1} are removed:



Lemma 2 – Injection idea



Each removed set I is mapped injectively to $I \setminus \{v\} \in \Omega(G_i)$, so

$$|\Omega(G_{i-1}) \setminus \Omega(G_i)| \leq |\Omega(G_i)|.$$



Lemma 2 – Proof (2)

Step 2: bias from almost-uniform sampling.

- Let μ be the probability that one sample from the $(\varepsilon/(6m))$ -uniform sampler for $\Omega(G_{i-1})$ lies in $\Omega(G_i)$.
- Since the sampler is $\varepsilon/(6m)$ -uniform,

$$|\mu - r_i| \leq \frac{\varepsilon}{6m}.$$

- From $r_i \geq 1/2$ we also get a relative bound

$$|\mu - r_i| \leq \frac{\varepsilon}{6m} \cdot 2 \cdot \frac{1}{2} \leq \frac{\varepsilon}{3m} r_i.$$

$$\text{and } \mu \geq r_i - \frac{\varepsilon}{6m} \geq \frac{1}{2} - \frac{\varepsilon}{6m} \geq \frac{1}{3}.$$



Lemma 2 – Proof (3)

Step 3: Chernoff bound for the sampling error.

- Let Y_1, Y_2, \dots, Y_M be independent indicators of the event “sample is in $\Omega(G_i)$ ”; then $X = \sum_{j=1}^M Y_j$, $\hat{r}_i = X/M$, and $\mathbb{E}[Y_j] = \mathbb{E}[\hat{r}_i] = \mu$.



Lemma 2 – Proof (3)

Step 3: Chernoff bound for the sampling error.

- Let Y_1, Y_2, \dots, Y_M be independent indicators of the event “sample is in $\Omega(G_i)$ ”; then $X = \sum_{j=1}^M Y_j$, $\hat{r}_i = X/M$, and $\mathbb{E}[Y_j] = \mathbb{E}[\hat{r}_i] = \mu$.
- By the Chernoff bound, for $0 < \delta \leq 1$,

$$\Pr[|\hat{r}_i - \mu| \geq \delta\mu] \leq 2 \exp\left(-\frac{\delta^2 \mu M}{3}\right).$$



Lemma 2 – Proof (3)

Step 3: Chernoff bound for the sampling error.

- Let Y_1, Y_2, \dots, Y_M be independent indicators of the event “sample is in $\Omega(G_i)$ ”; then $X = \sum_{j=1}^M Y_j$, $\hat{r}_i = X/M$, and $\mathbb{E}[Y_j] = \mathbb{E}[\hat{r}_i] = \mu$.
- By the Chernoff bound, for $0 < \delta \leq 1$,

$$\Pr[|\hat{r}_i - \mu| \geq \delta\mu] \leq 2 \exp\left(-\frac{\delta^2 \mu M}{3}\right).$$

- Use $\delta = \varepsilon/(8m)$ and our finding that $\mu \geq 1/3$, we have

$$\Pr\left[|\hat{r}_i - \mu| \geq \frac{\varepsilon}{8m} \mu\right] \leq 2 \exp\left(-\frac{\varepsilon^2 M}{576 m^2}\right).$$



Lemma 2 – Proof (3) (contd.)

$$\Pr\left[|\hat{r}_i - \mu| \geq \frac{\varepsilon}{8m} \mu\right] \leq 2 \exp\left(-\frac{\varepsilon^2 M}{576 m^2}\right).$$

- Our choice:

$$M = \left\lceil 1296 m^2 \varepsilon^{-2} \ln\left(\frac{2m}{\Delta}\right) \right\rceil \geq 576 m^2 \varepsilon^{-2} \ln\left(\frac{2m}{\Delta}\right)$$

guarantees

$$\Pr\left[|\hat{r}_i - \mu| \geq \frac{\varepsilon}{8m} \mu\right] \leq \frac{\Delta}{m}.$$



Lemma 2 – Proof (4)

Step 4: combine the two errors.

- On the “good” event from Step 3, $|\hat{r}_i - \mu| \leq \frac{\varepsilon}{8m} \mu$. (w.h.p.).
- Together with Step 2, $|\mu - r_i| \leq \frac{\varepsilon}{6m}$, and using $r_i \geq 1/2$ and μ close to r_i , a straightforward calculation shows

$$|\hat{r}_i - r_i| \leq \frac{\varepsilon}{2m} r_i.$$

- Therefore

$$\Pr\left[|\hat{r}_i - r_i| \leq \frac{\varepsilon}{2m} r_i\right] \geq 1 - \frac{\Delta}{m}.$$



Outline

- 1 Introduction
- 2 The DNF counting problem
- 3 DNF counting algorithms
- 4 Counting independent sets in a graph.
 - Markov Chain Monte Carlo (MCMC) Algorithms



FPAUS \rightarrow FPRAS

Theorem 3

Given an FPAUS for independent sets in any graph, we can construct an FPRAS for the number of independent sets in a graph G .

- Reason:
 - Use the FPAUS to approximately estimate each r_i .
 - Apply Lemma 2 for each r_i and Lemma 1 to control the product.
 - The number of samples M is polynomial in m , $1/\varepsilon$, and $\ln(1/\Delta)$, and each sample takes polynomial time.



How to get an FPAUS?

- We have reduced approximate counting to (almost) uniform sampling.



How to get an FPAUS?

- We have reduced approximate counting to (almost) uniform sampling.
- For independent sets, how do we obtain an FPAUS?



How to get an FPAUS?

- We have reduced approximate counting to (almost) uniform sampling.
- For independent sets, how do we obtain an FPAUS?
- One approach:
 - Use Markov Chain Monte Carlo (MCMC) with a carefully designed chain.
 - See coupling of Markov chains, e.g. Ch. 12 of [MU17].



Markov Chain Monte Carlo (MCMC)

- MCMC provides a general approach to sampling from a desired distribution.
- Basic idea:
 - Define an ergodic Markov chain whose **state space** is the **sample space**.
 - Ensure that its stationary distribution equals the desired distribution.
- Let X_0, X_1, \dots, X_n be a run of the chain.
- For a sufficiently large number of steps r , the distribution of X_r is **close to the stationary distribution** and can be used as a **sample**.



MCMC sampling

- Repeating the argument, we can use $X_r, X_{2r}, X_{3r}, \dots$ as (almost) independent samples from the stationary distribution.
- Efficiency of MCMC depends on:
 - How large r must be (**mixing time**).
 - Cost per step of the Markov chain.
- We focus on constructing efficient Markov chains with:
 - The **desired stationary distribution** (often uniform).
 - **Small mixing time** (for a rigorous FPAUS, we need mixing-time bounds).



Independent sets as states

- Let $G = (V, E)$ be a graph.
- State space: **all independent sets of G .**



Independent sets as states

- Let $G = (V, E)$ be a graph.
- State space: **all independent sets of G .**
- **Neighbor relation:**
 - Two states (independent sets) x and y are neighbors if they differ in exactly one vertex.
- This neighbor relation makes the state space **irreducible**.



Independent sets as states

- Let $G = (V, E)$ be a graph.
- State space: **all independent sets of G .**
- **Neighbor relation:**
 - Two states (independent sets) x and y are neighbors if they differ in exactly one vertex.
- This neighbor relation makes the state space **irreducible**.
 - Any independent set can be reached from the empty set by a sequence of vertex additions.
 - Conversely, we can delete vertices to reach the empty set.



Establish of the transition probabilities

- Next we need to establish the **transition probabilities**.
- A naïve approach: simple random walk on the state graph.



Establish of the transition probabilities

- Next we need to establish the **transition probabilities**.
- A naïve approach: simple random walk on the state graph.
- But this typically yields a stationary distribution where the probability of a state is **proportional to its degree** in the state graph.
- We need to modify the transition probabilities to get a uniform stationary distribution.



Towards a uniform stationary distribution

Lemma 3

- For a finite state space Ω with neighbor sets $N(x)$, if we choose transitions as

$$P_{x,y} = \begin{cases} 1/M, & x \neq y, y \in N(x), \\ 0, & x \neq y, y \notin N(x), \\ 1 - |N(x)|/M, & x = y, \end{cases}$$

for some $M \geq \max_{x \in \Omega} |N(x)|$, then, assuming irreducibility and aperiodicity, the stationary distribution is uniform.



Towards a uniform stationary distribution

Lemma 3

- For a finite state space Ω with neighbor sets $N(x)$, if we choose transitions as

$$P_{x,y} = \begin{cases} 1/M, & x \neq y, y \in N(x), \\ 0, & x \neq y, y \notin N(x), \\ 1 - |N(x)|/M, & x = y, \end{cases}$$

for some $M \geq \max_{x \in \Omega} |N(x)|$, then, assuming irreducibility and aperiodicity, the stationary distribution is uniform.

- That is, if we modify the random walk by giving each vertex an appropriate self-loop probability, then we can obtain a uniform stationary distribution.



Proof of Lemma 3

- For any $x \neq y$, since $\pi_x = \pi_y$ and $P_{x,y} = P_{y,x}$ ($= 1/M$), we have

$$\pi_x P_{x,y} = \pi_y P_{y,x}.$$

- Then, by applying the following theorem, it follows that the uniform distribution is the stationary distribution.

Theorem 4 (recall)

Consider a finite, irreducible, and ergodic Markov chain with transition matrix \mathbf{P} . If there are nonnegative numbers $\bar{\pi} = (\pi_0, \dots, \pi_n)$ such that $\sum_{i=0}^n \pi_i = 1$ and if, for any pair of states i, j ,

$$\pi_i P_{i,j} = \pi_j P_{j,i},$$

then $\bar{\pi}$ is the stationary distribution corresponding to \mathbf{P} .

Example chain for independent sets

- Consider the following simple Markov chain:
 - 1 X_0 is an arbitrary independent set in G .
 - 2 To get X_{i+1} from X_i :
 - 1 Choose a vertex $v \in V$ uniformly at random.
 - 2 If $v \in X_i$, set $X_{i+1} = X_i \setminus \{v\}$.
 - 3 If $v \notin X_i$ and $X_i \cup \{v\}$ is independent, set $X_{i+1} = X_i \cup \{v\}$.
 - 4 Otherwise, set $X_{i+1} = X_i$.
- The chain is irreducible and aperiodic.
- The stationary distribution can be shown to be uniform over independent sets, by applying Lemma 3.



Non-uniform stationary distributions

- Sometimes we want to sample according to a **non-uniform distribution**.
- Example: distributions where probability is proportional to some weight function.



Non-uniform stationary distributions

- Sometimes we want to sample according to a **non-uniform distribution**.
- Example: distributions where probability is proportional to some weight function.
- **Solution:** Use the **Metropolis algorithm**.



The Metropolis algorithm

- Given a connected state space Ω with neighbor structure $\{N(x) \mid x \in \Omega\}$.
- Desired stationary distribution:

$$\pi_x = \frac{b(x)}{B}, \quad B = \sum_{y \in \Omega} b(y), \quad b(x) > 0.$$

- The Metropolis chain:

$$P_{x,y} = \begin{cases} \frac{1}{M} \min\left(1, \frac{\pi_y}{\pi_x}\right), & x \neq y, y \in N(x), \\ 0, & x \neq y, y \notin N(x), \\ 1 - \sum_{z \neq x} P_{x,z}, & x = y, \end{cases}$$

for some $M \geq \max_{x \in \Omega} |N(x)|$.

- Under irreducibility and aperiodicity, the stationary distribution is π .



Correctness of the Metropolis algorithm

- For $x \neq y$ with $y \in N(x)$,

$$P_{x,y} = \frac{1}{M} \min\left(1, \frac{\pi_y}{\pi_x}\right), \quad P_{y,x} = \frac{1}{M} \min\left(1, \frac{\pi_x}{\pi_y}\right).$$

- For any $x \neq y$, suppose first that $\pi_x \leq \pi_y$. Then

$$P_{x,y} = \frac{1}{M}, \quad P_{y,x} = \frac{1}{M} \frac{\pi_x}{\pi_y}.$$

Hence

$$P_{x,y} = \frac{1}{M} = \frac{\pi_y}{\pi_x} P_{y,x} \implies \pi_x P_{x,y} = \pi_y P_{y,x}.$$

- The case $\pi_x > \pi_y$ can be similarly proved.
- For $x = y$ the equality $\pi_x P_{x,x} = \pi_x P_{x,x}$ is trivial.
- Thus, for all states x, y , $\pi_x P_{x,y} = \pi_y P_{y,x}$.



Example: independent sets with **weights**

- Let states be independent sets I in G .
- For a parameter $\lambda > 0$, define

$$\pi_x \propto \lambda^{|I_x|},$$

where I_x is the independent set of state x .

- That is,

$$\pi_x = \frac{\lambda^{|I_x|}}{\sum_y \lambda^{|I_y|}}.$$

- When $\lambda = 1$, this reduces to the uniform distribution.



Metropolis chain for weighted independent sets

- ① X_0 is an arbitrary independent set.
- ② To compute X_{i+1} from X_i :
 - ① Choose a vertex $v \in V$ uniformly at random (so $M = |V|$).
 - ② If $v \in X_i$, propose $Y = X_i \setminus \{v\}$ and accept with probability $\min(1, 1/\lambda)$.
 - ③ If $v \notin X_i$ and $X_i \cup \{v\}$ is independent, propose $Y = X_i \cup \{v\}$ and accept with probability $\min(1, \lambda)$.
 - ④ Otherwise, stay at X_i .
- For this chain, the stationary distribution is the desired $\pi_x \propto \lambda^{|I_x|}$.
- Note: we only ever need the ratios π_y/π_x , not the normalizing constant B .



Remarks

- For both uniform and weighted cases, we design a Markov chain whose stationary distribution is the one we want.
- Using coupling and mixing-time analysis (e.g., via ch. 12 in [MU17]), we can obtain an FPAUS.
- Combined with the reduction via ratios r_i , this yields FPRASs for certain counting problems.



Discussions

