Randomized Algorithms

The Monte Carlo Method

Joseph Chuang-Chieh Lin

Department of Computer Science & Engineering,
National Taiwan Ocean University

Spring 2026

Outline

Outline

Outline

- Introduction
  - The Monte Carlo Method
  - PRAS and FPRAS
- The DNF Counting Problem
- DNF counting algorithms
  - A naïve approach
  - An FPRAS
  - Approximate counting from FPAUS

Introduction

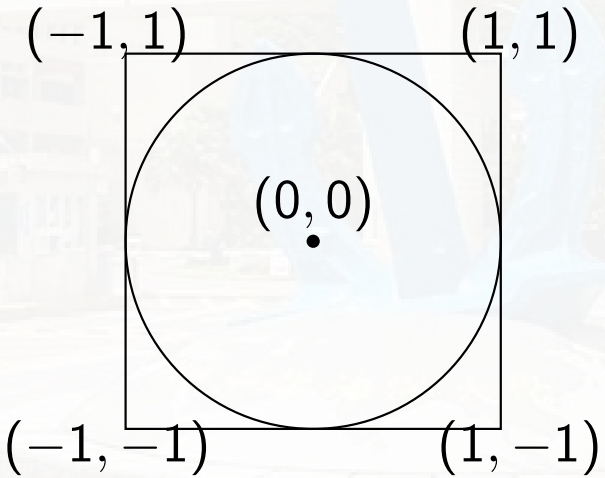
- The Monte Carlo method refers to a collection of tools for estimating values through sampling and simulation.
- Monte Carlo techniques are used extensively in almost all areas of physical sciences and engineering.

Introduction (example)

- As a motivating example, consider the following approach for estimating the constant π.
- The method is purely probabilistic and uses random sampling in a simple geometric setting.

Estimating π

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



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 $0 < \varepsilon < 1$,

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

- Thus, with enough samples, we obtain an arbitrarily good approximation to 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

$$\Pr\big[\,|X - V| \le \varepsilon V\,\big] \;\ge\; 1 - \Delta.$$

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

$$2e^{-m\pi\varepsilon^2/12} \le \Delta \quad \Longrightarrow \quad m \;\ge\; \frac{12}{\pi\varepsilon^2} \ln \frac{2}{\Delta}.$$

A general sampling theorem

- We generalize the idea behind estimating π to relate the number of samples and the approximation quality.
- Consider independent indicator random variables $X_1, \ldots, X_m$ with $\mu = \mathbb{E}[X_i]$.

Theorem 1. Let $X_1, \ldots, X_m$ 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 $P = NP$.
- 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 talk, 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.

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 $\mathrm{poly}(|x|)$.

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

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

Outline

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

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

  where each clause is a conjunction of literals.
- Example:

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

#P and #P-completeness

Class #P.

- A problem is in the class $\#\mathrm{P}$ if there is a polynomial-time nondeterministic Turing machine such that, for any input I, the number of accepting computations equals the number of solutions associated with I.
- Clearly, a $\#\mathrm{P}$ problem is at least as hard as the corresponding NP decision problem.

#P-complete.

- A problem is $\#\mathrm{P}$-complete if it is in $\#\mathrm{P}$ and every problem in $\#\mathrm{P}$ can be reduced to it in polynomial time.
- Examples:
  - Counting Hamiltonian cycles in a graph.
  - Counting perfect matchings in a bipartite graph.

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 and is in fact #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 $\#\mathrm{P}$-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


A naïve sampling algorithm

- Let $c(F)$ be 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, an approximation of $c(F)$.

1. $X \leftarrow 0$.
2. For $k = 1$ 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 $c(F) = \mathrm{poly}(n)$, 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

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

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

DNF counting algorithm II

DNF counting algorithm II (constructing S)

- **Input:** DNF formula F with n variables and clauses $C_1, \ldots, C_t$.
- **Output:** Y, an approximation of $c(F)$.

1. $X \leftarrow 0$.
2. For $k = 1$ to m:
   1. Choose $i \in \{1, \ldots, t\}$ 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 $a \in S_{C_i}$:

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

- 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 at least $1/t$.
- By Theorem 1, with

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

  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

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

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

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

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 $S \subseteq \Omega$,

$$\left|\Pr[w \in S] - |S|/|\Omega|\right| \leq \varepsilon.$$

FPAUS

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

From FPAUS to FPRAS (independent sets)

- Consider an FPAUS for independent sets.
- Input: a graph $G = (V, E)$ and parameter ε.
- Sample space: all independent sets in G.
- We want an FPRAS for $|\Omega(G)|$, the number of independent sets.

- Let $m = |E|$ and fix an ordering $e_1, \ldots, e_m$ 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 that G_0 has no edges, so $|\Omega(G_0)| = 2^{|V|}$.

Decomposition of $|\Omega(G)|$

- Define ratios

$$r_i = \frac{|\Omega(G_i)|}{|\Omega(G_{i-1})|}, \qquad i = 1, \ldots, m.$$

- Then

$$|\Omega(G)| = |\Omega(G_m)| = 2^n \prod_{i=1}^{m} r_i,$$

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

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

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.

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
$$M = \left\lceil 1296\, m^2 \varepsilon^{-2} \ln \frac{2m}{\Delta} \right\rceil$$
independent trials:
   1. Generate an $(\varepsilon/(6m))$-uniform sample from $\Omega(G_{i-1})$.
   2. 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 $\hat{r}_i$:

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

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

Theorem 3

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.
- 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. Chapter 11 of [MU05].

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 $X_0, X_1, \ldots, X_n$ 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 $X_r, X_{2r}, X_{3r}, \ldots$ 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.
- 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.

Towards a uniform stationary distribution

- 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.

Lemma 3 (informal).

- 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 $M \geq \max_x |N(x)|$, then, assuming irreducibility and aperiodicity, the stationary distribution is uniform.

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.
- **Solution:** Use the Metropolis algorithm.

The Metropolis algorithm

- Assume we have a connected state space Ω with neighbor structure $N(x)$.
- 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 $M \geq \max_x |N(x)|$.
- Under irreducibility and aperiodicity, the stationary distribution is π.

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 [MU05]), we can obtain an FPAUS.
- Combined with the reduction via ratios r_i, this yields FPRASs for certain counting problems.


Discussions