This lecture’s content is covered in Thomas Worsch’s notes from 2019.
1. What is Randomised Approximation?

2. Approximately counting satisfying assignments for Boolean formulas
Randomised Approximate Counting

Definition

A randomised algorithm $A$ approximates a quantity $f(x)$ if for any input $x$ the output $A(x)$ satisfies:

$$\Pr[|A(x) - f(x)| \leq \varepsilon \cdot f(x)] \geq 1 - \delta.$$ 

The parameters are the relative error $\varepsilon$ and the failure probability $\delta$.

Remark: Related Complexity Classes

PRAS. Problems admitting $A$ with running time polynomial in $|x|$, but not necessarily in $1/\varepsilon$ (for $\delta = 1/4$).

FPRAS. Problems admitting $A$ with running time polynomial in $|x|$ and $1/\varepsilon$ (for $\delta = 1/4$).

Note: Also defined where $f(x)$ is not a number. For instance: Want to compute a vertex cover with a size close to optimal.
Randomised Approximate Counting

Definition

A randomised algorithm \( A \) \emph{approximates} a quantity \( f(x) \) if for any input \( x \) the output \( A(x) \) satisfies:

\[
\Pr[|A(x) - f(x)| \leq \varepsilon \cdot f(x)] \geq 1 - \delta.
\]

The parameters are the relative error \( \varepsilon \) and the failure probability \( \delta \).

Only \( A(z) = 0 \) is acceptable if \( f(z) = 0 \)

\( \varepsilon = \frac{1}{4} \)
Randomised Approximate Counting

Definition

A randomised algorithm $A$ approximates a quantity $f(x)$ if for any input $x$ the output $A(x)$ satisfies:

$$\Pr[|A(x) - f(x)| \leq \varepsilon \cdot f(x)] \geq 1 - \delta.$$ 

The parameters are the relative error $\varepsilon$ and the failure probability $\delta$.

Remark: Related Complexity Classes

PRAS. Problems admitting $A$ with running time polynomial in $|x|$, but not necessarily in $\frac{1}{\varepsilon}$ (for $\delta = 1/4$).

FPRAS. Problems admitting $A$ with running time polynomial in $|x|$ and $\frac{1}{\varepsilon}$ (for $\delta = 1/4$).

Note: Also defined where $f(x)$ is not a number. For instance: Want to compute a vertex cover with a size close to optimal.
A counting problem

For Boolean formula \( B(x_1, \ldots, x_n) \) let \( \#B \) be the number of satisfying assignments:

\[
\#B = |\{(x_1, \ldots, x_n) \in \{0, 1\}^n \mid B(x_1, \ldots, x_n) = 1\}|.
\]

Example

\[
B = (x_1 \lor \overline{x}_2) \land (\overline{x}_1 \lor x_3)
\]

\[
\#B = |\{(0, 0, 0), (0, 0, 1), (1, 0, 1), (1, 1, 1)\}| = 4
\]
A counting problem

For Boolean formula $B(x_1, \ldots, x_n)$ let $\#B$ be the number of satisfying assignments:

$$\#B = |\{(x_1, \ldots, x_n) \in \{0, 1\}^n \mid B(x_1, \ldots, x_n) = 1\}|.$$

Example

$$B = (x_1 \lor \overline{x}_2) \land (\overline{x}_1 \lor x_3)$$

$$\#B = |\{(0, 0, 0), (0, 0, 1), (1, 0, 1), (1, 1, 1)\}| = 4$$


Assume $A$ satisfies $\Pr[|A(B) - \#B| \leq \varepsilon(\#B)] \leq 1 - \delta$ for $\varepsilon = \frac{1}{2}$ and $\delta = \frac{1}{4}$. Then

$$B \text{ is UNSAT} \iff \#B = 0 \Rightarrow \Pr[|A(B) - 0| \leq \frac{1}{2} \cdot 0] \geq \frac{3}{4} \Rightarrow \Pr[A(B) = 0] \geq \frac{3}{4}$$

$$B \text{ is SAT} \iff \#B > 0 \Rightarrow \Pr[|A(B) - \#B| \leq \frac{1}{2} \cdot \#B] \geq \frac{3}{4} \Rightarrow \Pr[A(B) > 0] \geq \frac{3}{4}$$

If $A$ is polynomial time then $A$ is BPP algorithm for SAT. Then SAT $\in$ BPP and NP $\subseteq$ BPP. Hard to believe...
What could be a tractable special case?

Relative error $\varepsilon < 1$ requires distinguishing:

$$\text{UNSAT } \Leftrightarrow \#B = 0 \quad \text{from} \quad \text{SAT } \Leftrightarrow \#B \geq 1.$$
What could be a tractable special case?

Relative error $\varepsilon < 1$ requires distinguishing:

$$\text{UNSAT} \iff \#B = 0 \quad \text{from} \quad \text{SAT} \iff \#B \geq 1.$$
What could be a tractable special case?

Relative error $\varepsilon < 1$ requires distinguishing:

$$\text{UNSAT} \iff \#B = 0 \quad \text{from} \quad \text{SAT} \iff \#B \geq 1.$$ 

An asymmetry for CNF formulas

- $B$ is called a tautology if $\#B = 2^n$.
- “is $B$ TAUT?” is easy to decide: Only empty CNF-formula is TAUT. (assuming $x_i$ and $\overline{x}_i$ never in the same clause)
- Try approximating unsatisfying assignments?

$$f(x) := 2^n - \#B.$$ 

CNF is hopeless

$$B = (x_1 \lor \overline{x}_2 \lor \overline{x}_{42}) \land \ldots \land (\overline{x}_1 \lor x_3 \lor \overline{x}_{37})$$

deciding SAT is NP-hard for clause size 3.
What could be a tractable special case?

Relative error $\varepsilon < 1$ requires distinguishing:

$\text{UNSAT} \iff \#B = 0$ from $\text{SAT} \iff \#B \geq 1$.

An asymmetry for CNF formulas

- $B$ is called TAUTology if $\#B = 2^n$.
- “is $B$ TAUT?” is easy to decide:
  - Only empty CNF-formula is TAUT.
  - (assuming $x_i$ and $\overline{x_i}$ never in the same clause)
- Try approximating unsatisfying assignments?
  - $f(x) := 2^n - \#B$.

Consider DNF!

$B' = \overline{B} = (\overline{x_1} \land x_2 \land x_{42}) \lor \ldots \lor (\overline{x_1} \land x_3 \land x_{37})$

- $\#B' = 2^n - \#B$.
- “$B'$ is SAT” is easy to decide
  - (only empty DNF-formula is UNSAT.)

CNF is hopeless

$B = (x_1 \lor \overline{x_2} \lor x_{42}) \land \ldots \land (\overline{x_1} \lor x_3 \lor x_{37})$

deciding SAT is NP-hard for clause size 3.
Intuition: Approximating $\pi$

Requirements for estimating area of disk (and hence $\pi$):
- Know formula for area of square
- Sample uniformly from square
- decide for $x, y \in [-1, 1]$ if $(x, y)$ in disk: $x^2 + y^2 \leq 1$

https://demonstrations.wolfram.com/ApproximatingPiByTheMonteCarloMethod/
Intuition: Approximating $\pi$

Requirements for estimating area of disk (and hence $\pi$):
- Know formula for area of square
- Sample uniformly from square
- decide for $x, y \in [-1, 1]$ if $(x, y)$ in disk: $x^2 + y^2 \leq 1$

https://demonstrations.wolfram.com/ApproximatingPiByTheMonteCarloMethod/
Approximate $|S|$ for $S \subseteq D$ by naive sampling

**Algorithm** $\text{approxSetSize}(1 \cdot \in S, D)$:

1. $\text{hits} \leftarrow 0$
2. **for** $i = 1$ to $N$ **do**
   1. sample $x \sim \mathcal{U}(D)$
   2. $\text{hits} \leftarrow \text{hits} + 1 \cdot x \in S$
3. **return** $\frac{\text{hits}}{N} \cdot |D|$

**Simple Theorem**

Let $D$ be a finite set and $S \subseteq D$ such that we can efficiently

- compute $|D|$
- sample uniformly from $D$
- decide for given $x \in D$ whether $x \in S$
Approximate $|S|$ for $S \subseteq D$ by naive sampling

**Algorithm** \( \text{approxSetSize}(1_{x \in S}, D) \):

\[
\begin{align*}
\text{hits} & \leftarrow 0 \\
\text{for } i = 1 \text{ to } N \text{ do} \\
\quad & \text{sample } x \sim \mathcal{U}(D) \\
\quad & \text{hits} \leftarrow \text{hits} + 1_{x \in S} \\
\text{return } \frac{\text{hits}}{N} \cdot |D|
\end{align*}
\]

**Simple Theorem**

Let \( D \) be a finite set and \( S \subseteq D \) such that we can efficiently

- compute \(|D|\)
- sample uniformly from \( D \)
- decide for given \( x \in D \) whether \( x \in S \)

Let \( p = \frac{|S|}{|D|} \). Then \( \text{approxSetSize} \) with \( N = \frac{3 \log(2/\delta)}{\varepsilon^2 p} \) approximates \(|S|\) with relative error \( \varepsilon \) and failure probability \( \delta \).

\( \leftrightarrow \) Special Case \( \varepsilon, \delta = \Theta(1) \): Need \( N = \Omega(1/p) \) samples.
Approximate $|S|$ for $S \subseteq D$ by naive sampling

Algorithm approxSetSize($\mathbb{1}_{x \in S}, D$):

1. hits ← 0
2. for $i = 1$ to $N$ do
   a. sample $x \sim \mathcal{U}(D)$
   b. hits ← hits + $1_{x \in S}$
3. return $\frac{\text{hits}}{N} \cdot |D|$

Simple Theorem

Let $D$ be a finite set and $S \subseteq D$ such that we can efficiently

- compute $|D|$
- sample uniformly from $D$
- decide for given $x \in D$ whether $x \in S$

Then approxSetSize with $N = \frac{3 \log(2/\delta)}{\varepsilon^2 p}$ approximates $|S|$ with relative error $\varepsilon$ and failure probability $\delta$.

$\rightarrow$ Special Case $\varepsilon, \delta = \Theta(1)$: Need $N = \Omega(1/p)$ samples.

Proof: Apply Chernoff to $\text{hits} \sim \text{Bin}(N, p)$.

\[
\Pr[\text{fail}] = \Pr[|\text{result} - |S|| > \varepsilon|S|] = \Pr[\left| \frac{\text{hits}}{N} \cdot |D| - |S| \right| > \varepsilon|S|] = \Pr[|\text{hits} - \frac{|S|}{|D|} N| > \varepsilon \frac{|S|}{|D|} N]
= \Pr[|\text{hits} - pN| > \varepsilon pN] = \Pr[|\text{hits} - \mathbb{E}[\text{hits}]| > \varepsilon \mathbb{E}[\text{hits}]] \leq 2 \exp(-\varepsilon^2 \mathbb{E}[\text{hits}]/3) = 2 \exp(-\varepsilon^2 pN/3) = \delta.
\]
Approximate $|S|$ for $S \subseteq D$ by naive sampling

Algorithm `approxSetSize(1.∈S, D)`:  
```
    hits ← 0
    for $i = 1$ to $N$ do
        sample $x \sim \mathcal{U}(D)$
        hits ← hits + $1_{x \in S}$
    return $\frac{\text{hits}}{N} \cdot |D|$
```

Simple Theorem

Let $D$ be a finite set and $S \subseteq D$ such that we can efficiently
- compute $|D|$
- sample uniformly from $D$
- decide for given $x \in D$ whether $x \in S$

Let $p = |S|/|D|$. Then `approxSetSize` with $N = \frac{3\log(2/\delta)}{\varepsilon^2 p}$ approximates $|S|$ with relative error $\varepsilon$ and failure probability $\delta$.

$\Leftrightarrow$ Special Case $\varepsilon, \delta = \Theta(1)$: Need $N = \Omega(1/p)$ samples.

Chernoff ($\rightarrow$ Concentration, slide 15)

For $\varepsilon \in (0, 1)$ and $X \sim \text{Bin}(N, p)$:

$$\Pr[|X - \mathbb{E}[X]| > \varepsilon \mathbb{E}[X]] < 2 \exp(-\varepsilon^2 \mathbb{E}[X]/3).$$

Application to $\#B$

- $S = \text{satisfying assignments of } B$
- $D = \{0, 1\}^n$
- $p = \frac{|S|}{|D|} = \frac{\#B}{2^n}$
- We may have $p = 1/2^n$
- $N = \Omega(2^n)$ required
- $\Rightarrow$
Of course this didn’t work
Did not exploit that $B$ is in DNF.
Approximating \( \#B \) for \( B \) in DNF

Assume \( B = C_1 \lor \ldots \lor C_m \)

where \( C_i \) contains \( \ell_i \) literals.

- \( D_i := \{ x \in \{0, 1\}^n \mid C_i(x) = 1 \} \) (satisfying assignments of \( C_i \))
- \( D := \{(i, x) \mid i \in [m], x \in D_i\} \) \( (= D_1 \cup \ldots \cup D_m) \)
- \( S := \{(i, x) \mid i \in [m], x \in D_i, x \notin D_1 \cup \ldots \cup D_{i-1}\} \)
Approximating $\#B$ for $B$ in DNF

Assume $B = C_1 \lor \ldots \lor C_m$ where $C_i$ contains $\ell_i$ literals.

- $D_i := \{x \in \{0, 1\}^n \mid C_i(x) = 1\}$ (satisfying assignments of $C_i$)
- $D := \{(i, x) \mid i \in [m], x \in D_i\}$ ($= D_1 \cup \ldots \cup D_m$)
- $S := \{(i, x) \mid i \in [m], x \in D_i, x \notin D_1 \cup \ldots \cup D_{i-1}\}$

**Observations**

- $|S| = \#B$
- $|D_i| = 2^{n-\ell_i}$ and we can efficiently sample from $\mathcal{U}(D_i)$:
  - set variables appearing in $C_i$ as required, others from $Ber(1/2)$.
- We can efficiently compute $|D| = \sum_{i=1}^m |D_i|$ and sample $(I, X) \sim \mathcal{U}(D)$:
  - First sample $I$ such that $\Pr[I = i] = \frac{|D_i|}{|D|}$.
  - Then sample $X \sim \mathcal{U}(D_i)$.
  - Yields $\Pr[(I, X) = (i, x)] = \frac{|D_i|}{|D|} \cdot \frac{1}{|D_i|} = \frac{1}{|D|}$ for all $(i, x) \in D$.
- We can efficiently decide “is $(i, x) \in S$?” (in time $O(mn)$)
- $p = \frac{|S|}{|D|}$ satisfies $p \geq \frac{1}{m}$.
Theorem

If $B$ is in DNF, then we can approximate $\#B$ in polynomial time (using $N = m \cdot \frac{3 \log(2/\delta)}{\varepsilon^2}$ samples) with relative error $\varepsilon$ and failure probability $\delta$. 

Takeaway
**Theorem**

If $B$ is in DNF, then we can approximate $\#B$ in polynomial time (using $N = m \cdot \frac{3}{\epsilon^2} \log(2/\delta)$ samples) with relative error $\epsilon$ and failure probability $\delta$.

**Intuition: Why did this work?**

**Naive strategy:**

Problem: $|S|/|\{0, 1\}^n|$ may be exponentially small

**Improved strategy:**

Advantage: $|S|/|D|$ is $\Omega(1/m)$. 

**Takeaway**
Conclusion

Randomised Approximation is Powerful

For $B$ in DNF:

- Computing $\#B$ exactly is $\#P$-complete.
- no deterministic approximation algorithm for such problems is known
- we analysed an efficient randomised approximation algorithm
Anhang: Mögliche Prüfungsfragen

- Was ist ein randomisierter Approximationsalgorithmus (für ein Zählproblem)?
- Wir haben das Zählproblem $\#B$ für Boolesche Formeln betrachtet. Hatten wir im allgemeinen Fall Erfolg? Warum nicht?
- Welchen Spezialfall haben wir uns vorgenommen? Wieso tritt dort nicht das selbe Problem auf wie im allgemeinen Fall?
- Wir haben einen Algorithmus gesehen der für zwei Mengen $S \subseteq D$ die Größe von $|S|$ schätzt.
  - Unter welchen Annahmen ist dieser anwendbar?
  - Wie hat der Algorithmus funktioniert?
  - Wie hängt die Anzahl der nötigen samples von $|S|$ und $|D|$ ab?
- Um $\#B$ für DNF Formel $B$ zu schätzen haben wir einen schlauerem Ansatz kennengerlernt.
  - Wie hat dieser funktioniert?
  - Wie vermeidet dieser das Problem das naiven Ansatzes?