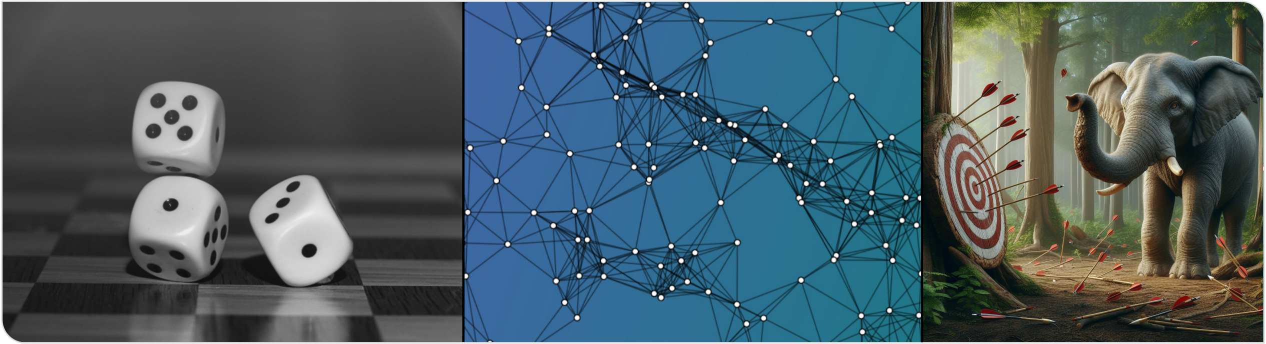
Karlsruhe Institute of Technology

Probability and Computing – Approximation Algorithms

Stefan Walzer | WS 2024/2025



Lecture Notes by Worsch

This lecture's content is covered in Thomas Worsch's notes from 2019.

Content

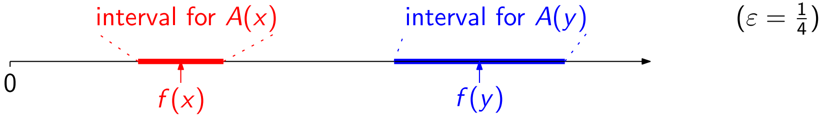
Randomised Approximate Counting

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* ε and the *failure probability* δ.



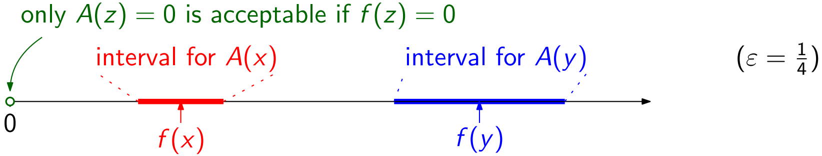
Randomised Approximate Counting

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* ε and the *failure probability* δ.



Randomised Approximate Counting

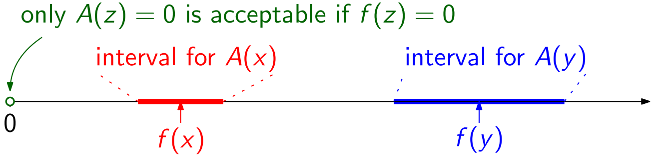
Karlsruhe Institute of Technology

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* ε and the *failure probability* δ.



$(\varepsilon = \frac{1}{4})$

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.

Counting Satisfiable Assignments of Boolean Formulas

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

$$B = (x_1 \vee \bar{x_2}) \wedge (\bar{x_1} \vee x_3)$$

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

Counting Satisfiable Assignments of Boolean Formulas

Karlsruhe Institute of Technology

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

$$B = (x_1 \vee \bar{x_2}) \wedge (\bar{x_1} \vee x_3)$$

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

Approximation algorithm for $\#B$ in general? Unlikely.

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

$$B \text{ is UNSAT} \Leftrightarrow \#B = 0 \Leftrightarrow \Pr[A(B) = 0] \geq \tfrac{3}{4}$$

$$B \text{ is SAT} \Leftrightarrow \#B > 0 \Leftrightarrow \Pr[A(B) > 0] \geq \tfrac{3}{4}$$

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?

We must distinguish: B is UNSAT $\Leftrightarrow \#B = 0$ from B is SAT $\Leftrightarrow \#B \geq 1$.

We need not distinguish: B is TAUT $\Leftrightarrow \#B = 2^n$ from B is NON-TAUT $\Leftrightarrow \#B < 2^n$.

What could be a tractable special case?

We must distinguish:	B is UNSAT $\Leftrightarrow \#B = 0$	from	B is SAT $\Leftrightarrow \#B \geq 1$.
We need not distinguish:	B is TAUT $\Leftrightarrow \#B = 2^n$	from	B is NON-TAUT $\Leftrightarrow \#B < 2^n$.

CNF is hard on the wrong end

$$(x_1 \vee \bar{x}_2 \vee \bar{x}_{42}) \wedge \ldots \wedge (\bar{x}_1 \vee x_3 \vee \bar{x}_{37})$$

Deciding UNSAT is NP-hard.
Deciding TAUT is easy.
$\hookrightarrow$ only empty CNF-formula is TAUT.

What could be a tractable special case?

We must distinguish: B is UNSAT $\Leftrightarrow \#B = 0$ from B is SAT $\Leftrightarrow \#B \geq 1$.

We need not distinguish: B is TAUT $\Leftrightarrow \#B = 2^n$ from B is NON-TAUT $\Leftrightarrow \#B < 2^n$.

CNF is hard on the wrong end

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

Deciding UNSAT is NP-hard.

Deciding TAUT is easy.

$\hookrightarrow$ only empty CNF-formula is TAUT.

DNF looks good

$$(\bar{x}_1 \land x_2 \land x_{42}) \lor \ldots \lor (x_1 \land \bar{x}_3 \land x_{37})$$

Deciding UNSAT is easy.

$\hookrightarrow$ only empty DNF-formula is UNSAT.

Deciding TAUT is hard.

What could be a tractable special case?

We must distinguish: B is UNSAT $\Leftrightarrow \#B = 0$ from B is SAT $\Leftrightarrow \#B \geq 1$.
We need not distinguish: B is TAUT $\Leftrightarrow \#B = 2^n$ from B is NON-TAUT $\Leftrightarrow \#B < 2^n$.

CNF is hard on the wrong end

$$(x_1 \vee \bar{x}_2 \vee \bar{x}_{42}) \wedge \ldots \wedge (\bar{x}_1 \vee x_3 \vee \bar{x}_{37})$$

Deciding UNSAT is NP-hard.
Deciding TAUT is easy.
↪ only empty CNF-formula is TAUT.

DNF looks good

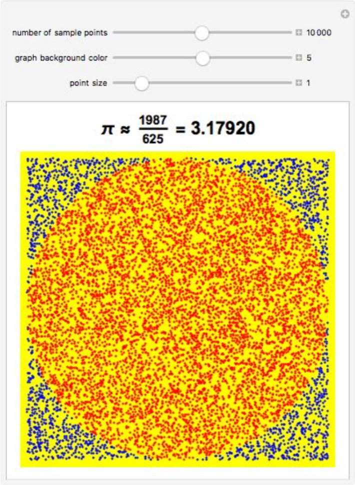
$$(\bar{x}_1 \wedge x_2 \wedge x_{42}) \vee \ldots \vee (x_1 \wedge \bar{x}_3 \wedge x_{37})$$

Deciding UNSAT is easy.
↪ only empty DNF-formula is UNSAT.
Deciding TAUT is hard.

Goal: Approximate $\#B$ for DNF formula B.

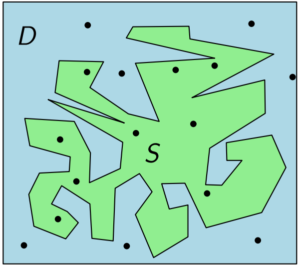
(Equivalently: Approximate $2^n - \#B$ for CNF formula B.)

Intuition: Approximating π



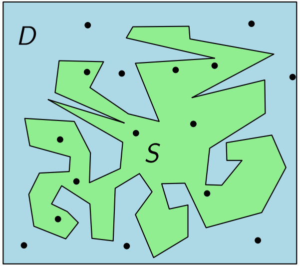
https://demonstrations.wolfram.com/ApproximatingPiByTheMonteCarloMethod/

Intuition: Approximate $|S|$ for $S \subseteq D$ by sampling from D



$|S| \approx |D| \cdot \frac{9}{16}$

Intuition: Approximate $|S|$ for $S \subseteq D$ by sampling from D



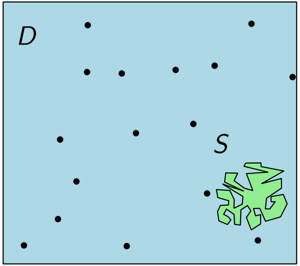
$$|S| \approx |D| \cdot \frac{9}{16}$$

Requirements

For this to work we must be able to

1. compute the size of D
2. sample uniformly from D
3. decide for $x \in D$ whether $x \in S$

Intuition: Approximate $|S|$ for $S \subseteq D$ by sampling from D



$$|S| \approx |D| \cdot \frac{0}{16}$$

Requirements

For this to work we must be able to

1. compute the size of D
2. sample uniformly from D
3. decide for $x \in D$ whether $x \in S$

soft requirement:

4. $\frac{|S|}{|D|}$ should not be too small

Approximate $|S|$ for $S \subseteq D$ by sampling from D

Algorithm approxSetSize**:**

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

Simple Theorem

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

1. compute $|D|$
2. sample uniformly from D
3. decide for $x \in D$ whether $x \in S$

Approximate $|S|$ for $S \subseteq D$ by sampling from D

Algorithm approxSetSize**:**

```
hits ← 0
for i = 1 to N do
    sample x ~ U(D)
    hits ← hits + 1_{x∈S}
return (hits/N) · |D|
```

Simple Theorem

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

1. compute $|D|$
2. sample uniformly from D
3. decide for $x \in D$ 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 ε and failure probability δ.

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

Approximate $|S|$ for $S \subseteq D$ by sampling from D

Algorithm approxSetSize**:**

$$
\begin{array}{l}
\text{hits} \leftarrow 0 \\
\textbf{for } i = 1 \textit{ to } N \textbf{ do} \\
\quad \text{sample } x \sim \mathcal{U}(D) \\
\quad \text{hits} \leftarrow \text{hits} + \mathbb{1}_{x \in S} \\
\textbf{return } \frac{\text{hits}}{N} \cdot |D|
\end{array}
$$

Simple Theorem

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

1. compute $|D|$
2. sample uniformly from D
3. decide for $x \in D$ 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 ε and failure probability δ.

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

Chernoff

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

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

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

$$\Pr[\text{fail}] = \Pr[\big|\text{result} - |S|\big| > \varepsilon|S|] = \Pr[\big|\tfrac{\text{hits}}{N} \cdot |D| - |S|\big| > \varepsilon|S|] = \Pr[\big|\text{hits} - \tfrac{|S|}{|D|}N\big| > \varepsilon\tfrac{|S|}{|D|}N]$$

$$= \Pr[\big|\text{hits} - pN\big| > \varepsilon pN] = \Pr[\big|\text{hits} - \mathbb{E}[\text{hits}]\big| > \varepsilon\mathbb{E}[\text{hits}]] \leq 2\exp(-\varepsilon^2\mathbb{E}[\text{hits}]/3) = 2\exp(-\varepsilon^2 pN/3) = \delta.$$

Approximate $|S|$ for $S \subseteq D$ by sampling from D

Algorithm approxSetSize:

```
hits ← 0
for i = 1 to N do
    sample x ~ U(D)
    hits ← hits + 1_{x∈S}
return hits/N · |D|
```

Simple Theorem

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

1. compute $|D|$
2. sample uniformly from D
3. decide for $x \in D$ 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 ε and failure probability δ.

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

Chernoff

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 $\#B$

- $S =$ 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
- :-(

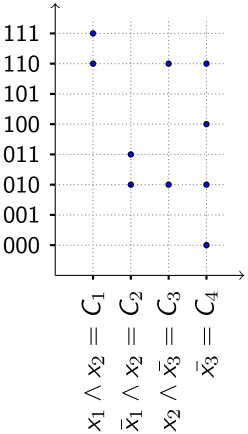
No Surprise

Of course this didn't work

Did not exploit that B is in DNF.

Approximating $\#B$ for B in DNF

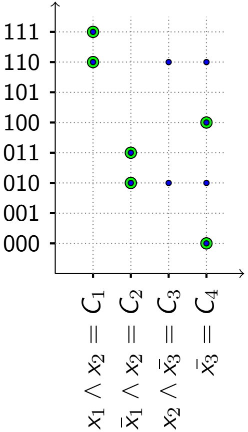
Assume $B = C_1 \vee \ldots \vee C_m$ where C_i contains ℓ_i literals.



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

Approximating $\#B$ for B in DNF

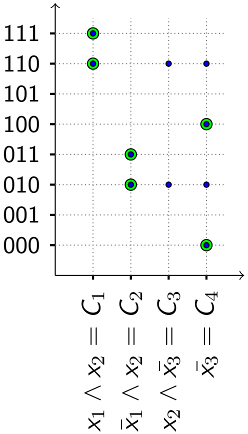
Assume $B = C_1 \vee \ldots \vee C_m$ where C_i contains ℓ_i literals.



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

Approximating $\#B$ for B in DNF

Assume $B = C_1 \vee \ldots \vee C_m$ where C_i contains ℓ_i literals.

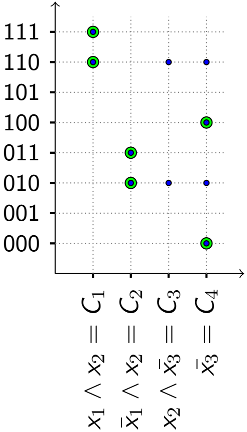


- $D_i := \{x \in \{0,1\}^n \mid C_i(x) = 1\}$ (satisfying assignments of C_i)
- $D := \{(i,x) \mid i \in [m], x \in D_i\}$ $(= D_1 \dot\cup \ldots \dot\cup D_m)$
- $S := \{(i,x) \mid i \in [m], x \in D_i, x \notin D_1 \cup \ldots D_{i-1}\}$

Claim: We can approximate $|S| = \#B$ by sampling from D

Approximating $\#B$ for B in DNF

Assume $B = C_1 \vee \ldots \vee C_m$ where C_i contains ℓ_i literals.



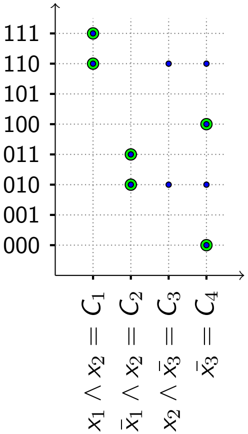
- $D_i := \{x \in \{0,1\}^n \mid C_i(x) = 1\}$ (satisfying assignments of C_i)
- $D := \{(i,x) \mid i \in [m], x \in D_i\}$ $(= D_1 \dot\cup \ldots \dot\cup D_m)$
- $S := \{(i,x) \mid i \in [m], x \in D_i, x \notin D_1 \cup \ldots D_{i-1}\}$

Claim: We can approximate $|S| = \#B$ by sampling from D

1. We can compute $|D_i| = 2^{n-\ell_i}$ and $|D| = \sum_{i=1}^m |D_i|$. ✓

Approximating $\#B$ for B in DNF

Assume $B = C_1 \vee \ldots \vee C_m$ where C_i contains ℓ_i literals.



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

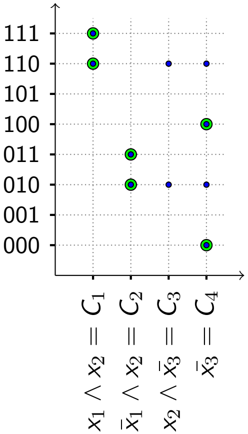
Claim: We can approximate $|S| = \#B$ by sampling from D

1. We can compute $|D_i| = 2^{n-\ell_i}$ and $|D| = \sum_{i=1}^{m} |D_i|$. ✓
2. We can efficiently sample $(I,X) \sim \mathcal{U}(D)$
   - sample I such that $\Pr[I = i] = \frac{|D_i|}{|D|}$
   - sample $X \sim \mathcal{U}(D_i)$
     $\hookrightarrow$ set variables appearing in C_i as required, sample others from $\mathrm{Ber}(1/2)$.
   - Yields $\Pr[(I,X) = (i,x)] = \frac{|D_i|}{|D|} \cdot \frac{1}{|D_i|} = \frac{1}{|D|}$ for all $(i,x) \in D$.

Approximating $\#B$ for B in DNF

Karlsruhe Institute of Technology

Assume $B = C_1 \vee \ldots \vee C_m$ where C_i contains ℓ_i literals.



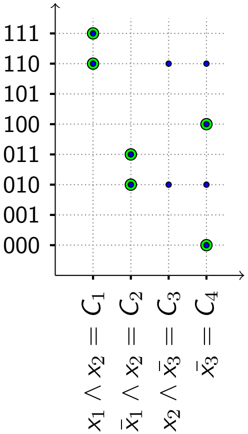
- $D_i := \{x \in \{0,1\}^n \mid C_i(x) = 1\}$ (satisfying assignments of C_i)
- $D := \{(i,x) \mid i \in [m], x \in D_i\}$ $(= D_1 \dot\cup \ldots \dot\cup D_m)$
- $S := \{(i,x) \mid i \in [m], x \in D_i, x \notin D_1 \cup \ldots D_{i-1}\}$

Claim: We can approximate $|S| = \#B$ by sampling from D

1. We can compute $|D_i| = 2^{n-\ell_i}$ and $|D| = \sum_{i=1}^{m} |D_i|$. ✓
2. We can efficiently sample $(I, X) \sim \mathcal{U}(D)$ ✓
   - sample I such that $\Pr[I = i] = \frac{|D_i|}{|D|}$
   - sample $X \sim \mathcal{U}(D_i)$
     $\hookrightarrow$ set variables appearing in C_i as required, sample others from $\mathrm{Ber}(1/2)$.
   - Yields $\Pr[(I,X) = (i,x)] = \frac{|D_i|}{|D|} \cdot \frac{1}{|D_i|} = \frac{1}{|D|}$ for all $(i,x) \in D$.
3. We can efficiently decide "is $(i,x) \in S$?"
   $\hookrightarrow$ just plug x into all clauses $C_1, \ldots, C_i$ // time $\mathcal{O}(mn)$

Approximating $\#B$ for B in DNF

Assume $B = C_1 \vee \ldots \vee C_m$ where C_i contains ℓ_i literals.



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

Claim: We can approximate $|S| = \#B$ by sampling from D

1. We can compute $|D_i| = 2^{n-\ell_i}$ and $|D| = \sum_{i=1}^m |D_i|$. ✓
2. We can efficiently sample $(I, X) \sim \mathcal{U}(D)$ ✓
   - sample I such that $\Pr[I = i] = \frac{|D_i|}{|D|}$
   - sample $X \sim \mathcal{U}(D_i)$
     $\hookrightarrow$ set variables appearing in C_i as required, sample others from $\text{Ber}(1/2)$.
   - Yields $\Pr[(I,X) = (i,x)] = \frac{|D_i|}{|D|} \cdot \frac{1}{|D_i|} = \frac{1}{|D|}$ for all $(i,x) \in D$.
3. We can efficiently decide "is $(i,x) \in S$?" ✓
   $\hookrightarrow$ just plug x into all clauses $C_1, \ldots, C_i$ // time $\mathcal{O}(mn)$
4. $p = \frac{|S|}{|D|}$ satisfies $p \geq \frac{1}{m}$. ✓

Takeaway

Karlsruhe Institute of Technology

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 ε and failure probability δ.

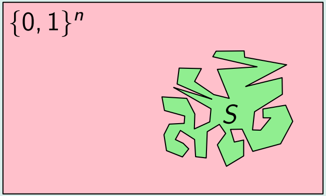
Takeaway

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 ε and failure probability δ.

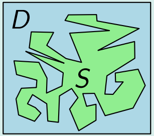
Intuition: Why did this work?

Naive strategy:



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

Improved strategy:



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

Conclusion

Randomised Approximation is Powerful

For B in DNF:

- Computing $\#B$ *exactly* is $\#\boldsymbol{P}$-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 schlaueren Ansatz kennengerlernt.
  - Wie hat dieser funktioniert?
  - Wie vermeidet dieser das Problem das naiven Ansatzes?