Counting Classes

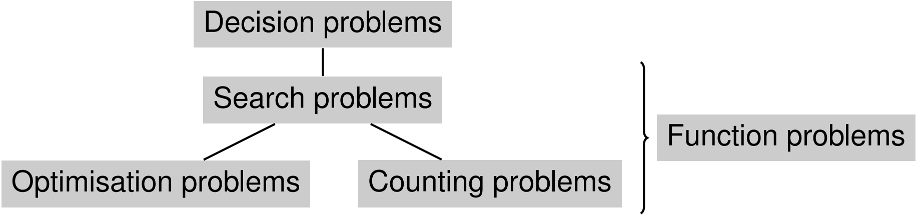
Complexity Theory

Faculty of Information Technology
Brno University of Technology
Brno, Czech Republic

Ondřej Lengál

This material was created with the support of the Czech Ministry of Education, Youth and Sports (project FRVŠ 166/2013/G1).

Classification of Problems



Given a relation $R \subseteq X \times Y$ and $x \in X$:

- Decision problems: decide membership in a language (yes/no).
  - Is there some $y \in Y$ s.t. $R(x, y)$?
- Function problems: generate some additional output.
  - **Search problems**: Find any $y \in Y$ s.t. $R(x, y)$.
  - **Optimisation problems**: Find the best $y \in Y$ s.t. $R(x, y)$.
  - **Counting problems**: How many $y \in Y$ are there s.t. $R(x, y)$?

Counting Problems

Definition (Counting problem)

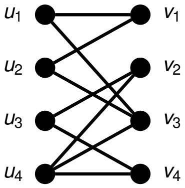
Consider a relation $R \subseteq X \times Y$ and the decision problem $D_R \subseteq X$ s.t. $x \in D_R \iff \exists y \in Y \,.\, R(x, y)$. The counting problem associated with R, $\#D_R$, is defined as

$$\#D_R(x) = |\{y \in Y \mid R(x, y)\}| \ .$$

Examples:

- $\#$SAT: how many different assignments satisfy given formula?
- $\#$CLIQUE: how many cliques of size k or larger are in a graph?
- $\#$HAMILTONIAN PATH: how many different Hamiltonian paths are in a graph?

Example: Counting Perfect Matchings



Definition (MATCHING)

Is there a perfect matching in the bipartite graph $G = (U, V, E)$?

Definition (#MATCHING)

How many perfect matchings are in the bipartite graph $G = (U, V, E)$?

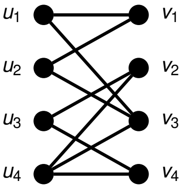
Example: Counting Perfect Matchings

Recall that MATCHING can be solved by checking whether the determinant of the adjacency matrix $\boldsymbol{A^G}$ of G is not identically zero.

$$\det \boldsymbol{A^G} = \sum_{\pi} \left(\sigma(\pi) \prod_{i=1}^{n} \boldsymbol{A}^{\boldsymbol{G}}_{i,\pi(i)} \right)$$

where

- π ranges over all permutation of n elements,
- $\sigma(\pi) = 1$ if π contains an even number of transpositions, else -1.



$$\boldsymbol{A^G} = \begin{array}{c|cccc} & v_1 & v_2 & v_3 & v_4 \\ \hline u_1 & 1 & 0 & 1 & 0 \\ u_2 & 1 & 0 & 1 & 0 \\ u_3 & 0 & 1 & 0 & 1 \\ u_4 & 0 & 1 & 1 & 1 \end{array}$$

Example: Counting Perfect Matchings

$$\det \boldsymbol{A^G} = \sum_{\pi} \left(\sigma(\pi) \prod_{i=1}^{n} \boldsymbol{A}^{\boldsymbol{G}}_{i,\pi(i)} \right)$$

- Note that the summation is done over all perfect matchings, but including the undesirable $\sigma(\pi)$ element.

Example: Counting Perfect Matchings

$$\det \boldsymbol{A}^{\boldsymbol{G}} = \sum_{\pi} \left(\sigma(\pi) \prod_{i=1}^{n} \boldsymbol{A}^{\boldsymbol{G}}_{i,\pi(i)} \right)$$

- Note that the summation is done over all perfect matchings, but including the undesirable $\sigma(\pi)$ element.
- If we get rid of the $\sigma(\pi)$ element, we arrive at a different characteristic of a matrix called the permanent.

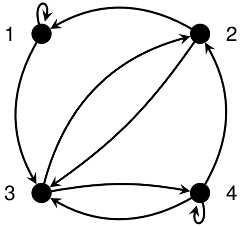
$$\operatorname{perm} \boldsymbol{A}^{\boldsymbol{G}} = \sum_{\pi} \left(\prod_{i=1}^{n} \boldsymbol{A}^{\boldsymbol{G}}_{i,\pi(i)} \right)$$

- The permanent of $\boldsymbol{A}^{\boldsymbol{G}}$ is precisely the number of perfect matchings in G, the problem is therefore known as PERMANENT.

Example: Counting Perfect Matchings

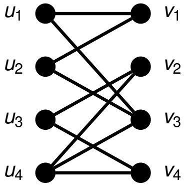
- Further, the number of perfect matchings in $G = (U, V, E)$ is equal to the number of cycle covers in the directed graph

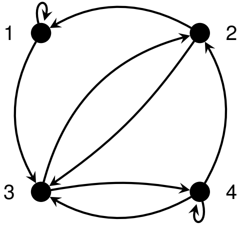
$$G' = (\{1, \ldots, |U|\}, \{(i,j) \mid (u_i, v_j) \in E\}) \, .$$



$$\boldsymbol{A^G} = \begin{array}{c|cccc} & v_1 & v_2 & v_3 & v_4 \\ \hline u_1 & 1 & 0 & 1 & 0 \\ u_2 & 1 & 0 & 1 & 0 \\ u_3 & 0 & 1 & 0 & 1 \\ u_4 & 0 & 1 & 1 & 1 \end{array}$$

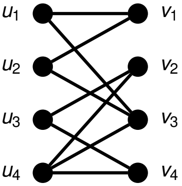
Example: Counting Perfect Matchings

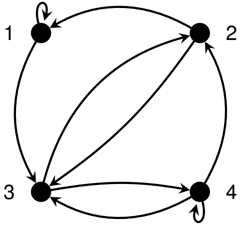




$$\boldsymbol{A}^{\boldsymbol{G}} = \begin{array}{c|cccc} & v_1 & v_2 & v_3 & v_4 \\ \hline u_1 & 1 & 0 & 1 & 0 \\ u_2 & 1 & 0 & 1 & 0 \\ u_3 & 0 & 1 & 0 & 1 \\ u_4 & 0 & 1 & 1 & 1 \end{array}$$

Example: Counting Perfect Matchings

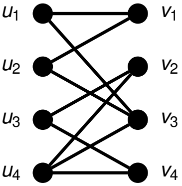


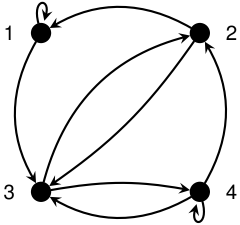


$$A^G = \begin{array}{c|cccc} & v_1 & v_2 & v_3 & v_4 \\ \hline u_1 & 1 & 0 & 1 & 0 \\ u_2 & 1 & 0 & 1 & 0 \\ u_3 & 0 & 1 & 0 & 1 \\ u_4 & 0 & 1 & 1 & 1 \end{array}$$

$$\operatorname{perm} \boldsymbol{A}^{\boldsymbol{G}} = \sum_{\pi} \left(\prod_{i=1}^{n} \boldsymbol{A}^{\boldsymbol{G}}_{i,\pi(i)} \right) = \ ?$$

Example: Counting Perfect Matchings





$$
\boldsymbol{A}^{\boldsymbol{G}} = \begin{array}{c|cccc}
 & v_1 & v_2 & v_3 & v_4 \\
\hline
u_1 & 1 & 0 & 1 & 0 \\
u_2 & 1 & 0 & 1 & 0 \\
u_3 & 0 & 1 & 0 & 1 \\
u_4 & 0 & 1 & 1 & 1
\end{array}
$$

$$
\operatorname{perm} \boldsymbol{A}^{\boldsymbol{G}} = \sum_{\pi} \left(\prod_{i=1}^{n} \boldsymbol{A}^{\boldsymbol{G}}_{i,\pi(i)} \right) = 4
$$

Example: Graph Reliability

Counting is relevant to probability; consider the decision problem

Definition (REACHABILITY)

Given a graph G, is there a path from node u to node v?

This gives rise to the following counting problem:

Definition (GRAPH RELIABILITY)

Given a graph G with m edges, how many of the 2^m subgraphs of G contain a path from node u to node v?

The problem is called GRAPH RELIABILITY because it gives a precise estimate of the probability that u and v will remain connected when all edges fail independently with probability $\frac{1}{2}$ each.

$\#$**P**

Definition ($\#$P)

$\#$**P** is the class of all counting problems associated with polynomially balanced polynomial-time decidable relations.

- $\#$**P** is pronounced "number **P**", "sharp **P**", or "pound **P**".
- Polynomially balanced relation: if $R(x, y)$, then $|y| \leq p(|x|)$.
- Polynomial-time decidable relation:
  - given x and y, it is checkable in polynomial time whether $R(x, y)$.

Reduction of Counting Problems

- All decision problems are easily reducible to their corresponding counting problems.
- As with other function problems, a reduction between counting problems A and B consists of two parts:
  - part R mapping instances x of A to instances $R(x)$ of B,
  - part S recovering from the answer y of $R(x)$ the answer $S(y)$ of x.
- For counting problems, there is a convenient class of reductions:

Definition (Parsimonious Reduction)

A reduction is parsimonious when $S = \mathrm{id}$.

$\#$SAT is $\#$**P**-complete

Theorem

$\#$SAT *is* $\#$**P***-complete.*

Proof.

Parsimonious variant of Cook's theorem (for CIRCUIT SAT):

- Each polynomially balanced and polynomial-time decidable binary relation $R \subseteq X \times Y$ together with $x \in X$ can be in deterministic polynomial time reduced to a CNF formula $\phi_{R(x)}$ with *input variables* $I = \{i_1, \ldots, i_n\}$.
- Each satisfying truth assignment to I corresponds to a unique $y \in Y$ s.t. $R(x, y)$. $\square$

PERMANENT is $\#\mathbf{P}$-complete.

Theorem (Valiant's Theorem)

PERMANENT *is* $\#\mathbf{P}$*-complete.*

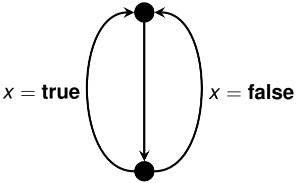
- Interesting because MATCHING $\in$ **P**.

Proof. (idea)

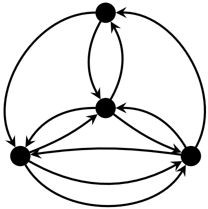
- By reduction from $\#$SAT.
- For a 3SAT formula ϕ, we construct a graph G_ϕ such that the cycle covers of G_ϕ somehow correspond to satisfying assignments of ϕ.
- The construction is very similar to the proof of **NP**-completeness of HAMILTONIAN PATH.

PERMANENT is #**P**-complete.

- For each Boolean variable x in ϕ, we create a choice gadget.
- For each clause in ϕ, we create a clause gadget:
  - no cycle cover traverses all 3 external edges,
  - for any proper subset S of external edges (including $\emptyset$), there is exactly one cycle cover traversing only external edges from S and no other external edges.



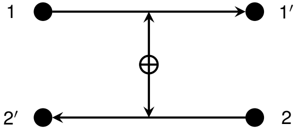
choice gadget



clause gadget

PERMANENT is $\#\mathbf{P}$-complete.

- External edges from clause gadgets are connected to corresponding edges of choice gadgets using XOR gadgets:
  - if exactly one of the edges $(1, 1')$ or $(2, 2')$ is traversed, the number of cycle covers is multiplied by 4,
  - there is no cycle cover in the graph if none or both are traversed.



XOR gadget

- For each satisfying assignment of ϕ, there are 4^m cycle covers
  - where m is the total number of literal occurrences in the formula.

- Details are rather technical and can be found in the literature:
  - structure of the XOR gadget,
  - reduction to PERMANENT MOD N.

How Strong Is Counting?

- Counting is very powerful indeed!
- Is $\#$**P** more powerful than **PH**?

How Strong Is Counting?

- Counting is very powerful indeed!
- Is $\#\mathbf{P}$ more powerful than **PH**?
- Note that we cannot directly compare $\#\mathbf{P}$ to **PH**:
  - $\#\mathbf{P}$ . . . a class of functions,
  - **PH** . . . a class of languages.
- However, recall the class **PP**:
  - **PP** . . . the class of languages L s.t. there is a poly. nondet. TM M, $x \in L$ iff more than $\frac{1}{2}$ computations of M on x end up accepting.
- There is a close relation between $\#\mathbf{P}$ and **PP**:
  - try looking at the MSB of the number of accepting computations.

Theorem (Toda's Theorem)

$$\mathbf{PH} \subseteq \mathbf{P}^{\mathbf{PP}}$$

$\oplus$**P**

Definition ($\oplus$P)

$\oplus$**P** is the class of languages L for which there is a polynomially balanced polynomial-time decidable relation R such that $x \in L$ iff the number of y's such that $R(x, y)$ is odd.

- $\oplus$**P** is pronounced "odd **P**", or "parity **P**".
- $\oplus$SAT and $\oplus$HAMILTONIAN PATH are $\oplus$**P**-complete,
  - a reduction similar to $\#$SAT and $\#$HAMILTONIAN PATH.

$\oplus$**P** $=$ **co**$\oplus$**P**

Theorem

$\oplus$**P** *is closed under complement, i.e.*

$$\oplus\mathbf{P} = \mathbf{co}\oplus\mathbf{P} .$$

Proof.

- The complement of $\oplus$SAT is obviously **co**$\oplus$**P**-complete.
- This language reduces to $\oplus$SAT of $\phi(x_1, \ldots, x_n)$ as follows:
  1. Add a new variable z to each clause of ϕ.
  2. Also add n clauses $(z \implies x_i)$ for $1 \leq i \leq n$.
- Any SAT assignment in the old formula is still SAT ($z =$ **false**).
- We get a new all-**true** SAT assignment ($z =$ **true**). $\square$

NP $\subseteq$ RP$^{\oplus \mathbf{P}}$

Theorem

$$\mathbf{NP} \subseteq \mathbf{RP}^{\oplus \mathbf{P}}$$

- **RP** . . . the class of languages for which there exists a polynomial Monte Carlo Turing machine.

Proof. (idea)

- Construct a polynomial MC TM for SAT using an oracle for $\oplus$SAT.
- We are given formula ϕ over variables $\{x_1, \ldots, x_n\}$.
- For $S \subseteq \{x_1, \ldots, x_n\}$ a hyperplane η_S is a Boolean expression in CNF stating an even number among the variables in S are **true**.
  - For variables $y_0, \ldots, y_n$, η_S is the conjunction of clauses (y_0), (y_n),

$$\text{plus for each } 1 \leq i \leq n \quad \begin{cases} (y_i \iff (y_{i-1} \oplus x_i)) & \text{if } x_i \in S \\ (y_i \iff y_{i-1}) & \text{if } x_i \notin S \end{cases}$$

$\mathbf{NP} \subseteq \mathbf{RP}^{\oplus\mathbf{P}}$

- The algorithm:
  1. $\phi_0 := \phi$
  2. For $i = 1, \ldots, n+1$ repeat the following:
     1. Generate a random subset $S_i \subseteq \{x_1, \ldots, x_n\}$.
     2. Set $\phi_i = \phi_{i-1} \wedge \eta_{S_i}$.
     3. If $\phi_i \in \oplus$SAT answer "ϕ is satisfiable."
     4. Else continue.
  3. Answer "ϕ is probably unsatisfiable."

- The probability of a false negative is no larger than $\frac{7}{8}$.
  - becomes less than $\frac{1}{2}$ by repeating the algorithm $6\times$. □