Lec 9. Counting Problems

#project-hardness

Changki Yun (TAMREF)
tamref.yun@snu.ac.kr

Seoul National University

December 6, 2022

Definition of $\sharp\mathbb{P}$

- We say $\phi \in \mathrm{SAT}$ if there's an assignment x such that $\phi(x)$ is true.
- Indeed, we are interested in the number of such x's.
- The function $\sharp\mathrm{SAT}$ counts the number of satisfying assignments of given CNF-formula ϕ.
- Does $\sharp\mathrm{SAT}$ belongs to $\mathbb{FP}$? Unlikely.

Hardness of $\sharp\mathrm{SAT}$

Toda's theorem (1991)

$\mathbb{PH} \subseteq \mathbb{P}^{\sharp\mathrm{SAT}}$. In other words, QBF is polynomial-time tractible with constant-time oracles on $\sharp\mathrm{SAT}$.

Thus we guess $\sharp\mathrm{SAT}$ is strictly harder than SAT, as $\mathbb{P}^{\mathrm{SAT}} = \Sigma_2 \subseteq \mathbb{PH} = \bigcup_{i=0}^{\infty} \Sigma_i$.

- Now it is natural to pick $\sharp\mathrm{SAT}$ as the reference problem of hard "counting problems".

Counting version of $\mathbb{NP}$-problems

We naturally extend this approach to a general decision problem $A \in \mathbb{NP}$. There is a polynomial p and set $B \in \mathbb{P}$ such that

$$A = \{\phi : \exists x \text{ s.t. } |x| = p(|\phi|) \wedge (\phi, x) \in B\}.$$

The counting function $\sharp A$ takes input ϕ, returning the number of x's with length $p(|\phi|)$, and $(\phi, x) \in B$.

Warning

Indeed such p, B are not defined uniquely for a decision problem A. We pick such p, B "naturally" for any A.

Definition & Reductions in $\sharp\mathbb{P}$

We define the class $\sharp\mathbb{P}$ to be $\{\sharp A : A \in \mathbb{NP}\}$.

Reductions on computable functions

Given function f, g.

- We denote $f \leq_{p.o.} g$ if there is a polynomial time algorithm $M(O_g, x)$ to compute $f(x)$, equipped with a constant-time oracle O_g to compute g.

Theorem. For any $\sharp A \in \sharp\mathbb{P}$, $\sharp A \leq_{p.o.} \sharp\mathrm{SAT}$.
We call f is $\sharp\mathbb{P}$-hard if $\sharp\mathrm{SAT} \leq_{p.o.} f$, and is $\sharp\mathbb{P}$-complete if $f \in \sharp\mathbb{P}$ additionally.

Parsimonious reductions

Some kind of reductions in $\mathbb{NP}$ are special, that preserving the number of solutions.

Parsimonious reductions

Given $A, B \in \mathbb{NP}, f : A \longrightarrow B$ is called parsimonious if both are satisfied.

1. $x \in A \iff f(x) \in B$. (reduction)
2. $\sharp A(x) = \sharp B(f(x))$.

If $f : A \longrightarrow B$ is parsimonious, both trivially holds.

- If $\sharp A$ is $\sharp P$-complete, so is $\sharp B$.
- If $\sharp B \in \mathbb{FP}$, so is $\sharp A$.

ASP problems

Author's old affection on "puzzle design" introduces another kind of decision problem – the **Another Solution Problems (ASP)**.

Another Solution Problem

Given $A \in \mathbb{NP}$ and input x equipped with a solution y such that $(x, y) \in B$. Is there another solution $y' \neq y$ satisfying $(x, y') \in B$? Such problem is called $\mathrm{ASP} - A$.

We have a sudoku puzzle x with an "intended" solution y, willing to prevent the "unintended" one y'. According to this desire, we don't need to actually determine y'.

ASP problems

$\mathrm{ASP} - A$ is not guaranteed to inherit the hardness of A.

- $\mathrm{ASP} - 3COL \in \mathbb{P}$, as we can just permute the colors.
- For cubic graphs, there are always even number of hamiltonian cycles. *(Tutte, 1946)* Thus $\mathrm{ASP} - CUBIC - HAM - CYCLE \in \mathbb{P}$, even though $CUBIC - HAM - CYCLE$ is $\mathbb{NP}$-complete.

These kind of reductions are useful to solve ASP-problems mechanically.

ASP-reduction

Given $A, B \in \mathbb{NP}$, Reduction $f : A \longrightarrow B$ is called ASP-reduction if f is parsimonious, and there's an (polynomial-time) algorithmic bijection ρ from a solution (x, y) of A to $(f(x), \rho(y))$ of B.

As it guarantees ASP-B is harder than ASP-A.

Hardness of $\sharp A$

- We'd like to assert "if A is $\mathbb{NP}$-complete, then $\sharp A$ is $\sharp\mathbb{P}-\text{complete}$ as well.".
    - True in general, but hard to write in theorem – since $\sharp A$ is defined on some vague "naturalness".
- Indeed there are cases for $A \in \mathbb{P}$, its counting version $\sharp A$ is $\sharp\mathbb{P}-\text{complete}$.

Hardness of $\sharp A$

Hard counting variants of easy decision problems

The natural "counting variants" of these "easy problems" are $\sharp P$-complete.

1. **(Brightwell & Winkler, 2005)** Given a graph, detect an eulerian cycle.
2. **(Jerrum, 1994)** Given an undirected graph, detect a tree (of any size) as a subgraph.
   - Counting is hard even for planar graphs.
3. **(Valiant, 1979)** Given a bipartite graph, detect a perfect matching as a subgraph.
   - Hard for some restricted kind of graphs, as planar / k-regular.
4. **(Valiant, above paper)** Given positive-literal only 2-CNF formula, find a satisfying solution.

Canonical Examples

Now we deduce hardness results from parsimonious reductions, starting from $\sharp\mathrm{SAT}$.

Parsimonious Reductions

There is a series of parsimonious reductions from $\sharp\mathrm{SAT}$, going through $\sharp 3\mathrm{SAT}$, $\sharp 3\mathrm{SAT} - 3$, and $\sharp\mathrm{CLIQUE}$. Moreover, all the reductions appear are ASP.

$\sharp$SAT $\to$ $\sharp$3SAT

Suppose $\phi = C_1 \wedge \cdots \wedge C_k$ where C_i are all OR-clauses.

- Introduce three auxiliary "false representer" variables F_1, F_2, F_3 and 7 additional clauses, all possible combinations of 3-clauses but $F_1 \vee F_2 \vee F_3$, so that any satisfying assignment has to set F_i to false.
- Compensate C_i's with < 3 literals, by adding F_1 and F_2.

$\sharp$SAT $\rightarrow$ $\sharp$3SAT **(Cont'd)**

- For C_i's with ≥ 4 literals, we parsimoniously decrease the number of literals in C_i, by adding "NAND" of the first two literals.
- For example, suppose $C_i = L_1 \vee L_2 \vee L_3 \vee L_4 \vee L_5$. We add a variable N_{12}, to formulate the equivalent one

$$\begin{aligned}
&(L_1 \vee L_2 \vee N_{12}) \wedge (L_1 \vee \neg L_2 \vee N_{12}) \\
&\wedge (\neg L_1 \vee L_2 \vee N_{12}) \wedge (\neg L_1 \vee \neg L_2 \vee \neg N_{12}) \\
&\wedge (N_{12} \vee L_3 \vee L_4 \vee L_5)
\end{aligned}$$

without changing the number of solutions. $\square$

$\sharp 3\mathrm{SAT} \to \sharp 3\mathrm{SAT} - 3$

If a variable x occurs $m > 3$ times, we just replace its occurrence with new variables $x_1, \cdots, x_m$. And add clauses $x_1 \to x_2, x_2 \to x_3, \cdots, x_m \to x_1$ to guarantee the equivalence. Note that it does not change the number of solutions. $\square$

$\sharp$3SAT $\rightarrow$ $\sharp$CLIQUE

- Assume $\phi = C_1 \wedge \cdots \wedge C_k$. Make a graph G_ϕ with $7k$ vertices, 7 vertices per a clause.
- For each clause, each vertices is associated with a satisfying assignments of the clause. For example, $\neg x \vee y \vee z$ generates 7 assignment-vertices except $(x = 1, y = 0, z = 0)$.
- Edges in G_ϕ are drawn between non-conflicting assignment variables. Now, ϕ is in $3\mathrm{SAT}$ iff G_ϕ has a k-CLIQUE, and each satisfying assignment and k-CLIQUE are one-to-one corresponded. $\square$

Planar Variants

Most of the reductions, done by fancy gadgets are parsimonious – need to be proven though.

- $\sharp\mathrm{PLANAR}-\mathrm{RECTLINEAR}-3\mathrm{SAT}$ is $\sharp\mathbb{P}-\mathrm{complete}.$
- $\sharp\mathrm{PLANAR}-1-\mathrm{IN}-3\mathrm{SAT}$ is $\sharp\mathbb{P}-\mathrm{complete}$, while the reduction in text is not parsimonious.
- $\sharp\mathrm{PLANAR}-\mathrm{HAM}-\mathrm{CYCLE}-\mathrm{MAX}-\mathrm{DEG}-3$ is $\sharp\mathbb{P}-\mathrm{complete}.$
- $\sharp\mathit{SLITHERLINK}$ is $\sharp\mathbb{P}-\mathrm{complete}.$

Hardness of Permanent

Historically, $\sharp\mathbb{P}$-completeness of **Permanent** is the most important result for the early $\sharp\mathbb{P}$-hardness. Note that permanent of a non-negative integer matrix M corresponds to the number of perfect bipartite matching.

$$\mathrm{per}(M) := \sum_{\sigma} M_{1,\sigma_1} \cdots M_{n,\sigma_n}$$

Many believed that permanent belongs to $\mathbb{P}$, reducing to determinant question with some clever edge-orientation.
Even though it was discovered to false, the class of graphs "properly orientable" had its own meaning.

Cycle-cover view of permanent

Viewing M as an adjacency matrix of a weighted graph, define "cycle cover" of a graph.

Cycle cover

A **cycle cover** is a set of n edges composing disjoint cycles.
Given a cycle cover C, its weight $w(C)$ is defined by the product of edge-weights in C.

Then $\mathrm{per}(M) = \sum_C w(C)$.

Proof sketch for permanent hardness

Several proofs from Valiant1979, BH1993, and Aar2011 are presented for $\sharp\mathbb{P}-\text{completeness}$ of Permanent, but the proofs are unintuitive and sophisticated.

- Whenever no input edge and no output edge of the component are used, the sum over all completions in it must equal 0. This can be represented by the equation

$$Perm(A) = 0$$

It can be verified that the following 7×7 matrix satisfies all these conditions. As the matrix A satisfies the above constraints, the clause component indeed has the properties we use in the proof of lemma 2.

$$A = \begin{pmatrix} 1 & 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 1 & 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & -1 & 1 & 1 \\ 0 & 0 & -1 & 2 & -1 & 1 & 1 \\ 0 & 0 & -1 & -1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 2 & -1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 \end{pmatrix}$$

Notably, Aar2011 involves quantum computation in proof. I will write a blog if my time allows.

Proof sketch for permanent hardness

Given a 3SAT instance, we convert variables into variable-vertices, and clauses into **clause-gadgets.**

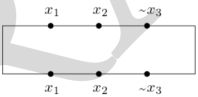

Figure 12.7: Gadget for Clause $(x_1 \vee x_2 \vee \neg x_3)$

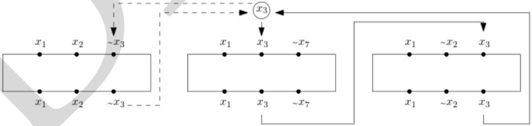

Figure 12.8: Gadget for x_3 in formula $(x_1 \vee x_2 \vee \neg x_3) \wedge (x_1 \vee x_3 \vee \neg x_7) \wedge (x_1 \vee \neg x_2 \vee x_3)$

There's a sophisticated structure inside the box, but basically it has 3 input vertices and 3 output vertices.

Proof sketch for permanent hardness

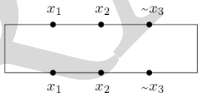

Figure 12.7: Gadget for Clause $(x_1 \vee x_2 \vee \neg x_3)$

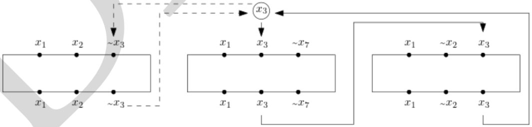

Figure 12.8: Gadget for x_3 in formula $(x_1 \vee x_2 \vee \neg x_3) \wedge (x_1 \vee x_3 \vee \neg x_7) \wedge (x_1 \vee \neg x_2 \vee x_3)$

Starting from each variable vertex x_i, it goes through all x_i-vertices and returns. And another cycle goes through all $\neg x_i$-vertices and returns.
Then, for each cycle covers, cycles containing the variable vertex determine the assignment of SAT problem.

Proof sketch for permanent hardness

With a delicated design, weight of each cycle cover C is

$$w(C) = \begin{cases} 12^m & \text{corresponding assignment is satisfying} \\ 0 & \text{otherwise} \end{cases}$$

So given a 3SAT instance ϕ and its derived graph G,
$\mathrm{per}(\mathrm{adj}(G)) = 12^m \sharp\mathrm{SAT}(\phi)$. $\square$
Moreover, we can obtain hardness for even $0-1$ matrices.

Permanent modulo r

For general integer m, $\mathrm{per}(M) \mod m$ is hard as well, even for $0-1$ matrices.
As $\mathrm{per}(M) \le n!$, collecting $O(n \log n)$ primes and compute $\mathrm{per}(M)$ modulo such primes and collect it by CRT.
For fixed r, Valiant obtained the following result.

- If $r = 2^d$, there is a $O(n^{4d-3})$ algorithm to compute $\mathrm{per}(M) \mod r$.
- If $r \neq 2^d$ for any d, $\mathrm{per}(M) \mod r$ is $\mathbb{UP}$-hard.

Counting Matchings is Hard

- We know that counting bipartite **perfect** matching is $\sharp\mathbb{P}$–complete.
- We show that even counting bipartite **maximal matching** is hard. Note that finding a maximal matching is enough with greedy algorithm.
- Denote each by $\sharp\mathrm{BPM}$, and $\sharp\mathrm{BMM}$.

Counting Matchings is Hard

The simple idea is weighting all maximal matchings by its cardinality. Given a graph G, replace $v \in V(G)$ by n^2 replicas, and edge $(v, w) \in E(G)$ by K_{n^2,n^2}.

Call this extended graph $\widetilde{G}$. If the number of maximal matchings in G with k edges are denoted to m_k,

$$\sharp\mathrm{BMM}(\tilde{G}) = \sum_{i=0}^{n} m_i \cdot (n^2!)^i.$$

Note that $m_i \leq \binom{n^2}{i} < n^2!$, we just take $m_n = \sharp\mathrm{BPM}(G)$ from base $n^2!$-expansion.

Counting SAT is mostly hard

Valiant deduced that even $\sharp 2\mathrm{SAT}$ is $\sharp\mathbb{P}-\mathrm{complete}$, and Nadia 1996 proved that there's a dichotomy theorem for SAT-counting; every constraints should be affine to be polynomially countable.

$\sharp\mathbb{P}-\mathrm{complete}$-SAT type problems

It is $\sharp\mathbb{P}-\mathrm{complete}$ to count satisfying assignments to the following class of problem.

- true-satisfiable. (positive literals only)
- bijuntive. (2SAT)
- Horn-SAT.

TH-POS-2SAT

The following problem is intriguing.

THreshold-POSitiveonly-2SAT

The problem takes input (ϕ, t), a CNF formula ϕ consisted of positive literals and the integer $t \geq 0$. Can you find an assignment with $\geq t$ variables set to false?

Note that ϕ is trivially satisfiable by setting all variables TRUE.

TH-POS-2SAT

There is a parsimonious reduction $PM \to TH - POS - 2SAT$. Note that it does not say anything about TH-POS-2SAT itself.

- Given a graph G with $2k$ vertices. For each edge e, assign variable x_e. For each incident edges e, f, add a clause $x_e \vee x_f$. Denote the obtained CNF as ϕ_G
- Then there's a perfect matching G iff there's a satisfying TH-POS-2SAT instance for (ϕ_G, k).
- It can be easily shown that the reduction is parsimonious, as the bijection is apparent.