CMSC 858M: Fun with Hardness
Spring 2021

Instructor: Mohammad T. Hajiaghayi
Scribe: Erika Melder

April 7, 2021

1 Overview

In this chapter, we define the class of counting problems #P, which contains all functions producing the number of solutions to some instance of a problem in NP. We analyze the hardness of this class. We then discuss the notion of parsimonious and c-monious reductions, which are reductions preserving the number of solutions or a multiple thereof, respectively, between the two problem instances in the reduction. We look at counting versions of various problems, and perform parsimonious reductions to establish their membership in #P, #P-completeness, or hardness relative to #P. Finally, we discuss the Another Solution Problem (ASP), which involves looking at a witness for some problem in NP and determining if another solution exists, and then define ASP-reducibility.

2 #SAT and the Class #P

The problem SAT asks whether a given Boolean formula φ has some satisfying assignment. We can define the counting version of this problem, denoted #SAT, as the problem of returning the *number* of satisfying assignments for a Boolean formula φ. We can similarly define counting versions of any problem in NP.

Definition 1 *Consider a set $A \in NP$. Then there exists a polynomial p and a set $B \in P$ such that*

$$A = \{x : (\exists y)[|y| = p(|x|) \wedge (x, y) \in B]\}.$$

#A is the function which, on input x, returns the number *of y of length $p(|x|)$ with $(x, y) \in B$.*

This definition is not unique, since there may be multiple choices of p and B creating arbitrary conditions on our witnesses. We therefore assume that the choice of p and B are natural with respect to the problem statement.

We define the set of all such counting problems as follows.

Definition 2 *#P is the set of all functions of the form #A for* $A \in NP$.

It is likely that this category of problems is harder than their corresponding decision problems. In particular, Toda proved that every problem in the polynomial hierarchy is reducible to #3SAT (using potentially many evaluations of #3SAT) [**?**]. This would imply that, if #3SAT is computable in polynomial time, then the entire polynomial hierarchy would be contained in P.

3 Reductions and #P-Completeness

Recall that if A is polynomially reducible to B (i.e. $A \leq_p B$), then we can solve whether $x \in A$ with a single call to B, where the answer to that call is true if and only if $x \in A$. We can expand the notion of reducibility to one which can use many calls in many ways via an oracle Turing machine that can query B.

Definition 3 *For functions* f *and* g*, we write* $f \leq_{p,o} g$ *if there is some oracle Turing machine* $M^{()}$ *such that* $f(x) = M^g(x)$ *and* $M^g(x)$ *can be calculated in polynomial time.*

This allows us to define a notion of #P-hardness, and consequently #P-completeness.

Definition 4

- *A function* g *is* #P-hard *if for all* $f \in \#P$, $f \leq_{p,o} g$.
- *A function* g *is* #P-complete *if* $g \in \#P$ *and* g *is #P-hard.*

In practice, however, we often show #P-completeness using a special type of reduction on the non-counting versions of the respective problems.

Definition 5 *Let* A, B *be sets in NP. A* c-monious reduction *from* A *to* B *is a reduction* f *satisfying both of the following:*

1. $x \in A \Leftrightarrow f(x) \in B$
2. *There exists a constant* $a \in \mathbb{Z}$ *such that the number of witnesses asserting* $x \in A$ *is equal to* a *times the number of witnesses asserting* $f(x) \in B$.

When $a = 1$ *(i.e. the two instances have equal numbers of solutions), this is known as a* parsimonious reduction.

We can establish #P-completeness of a counting problem #A by finding a parsimonious reduction from the decision version A to some other decision problem B whose counting version #B is #P-complete.

Scribe: Erika Melder
Chapter 13 **Date:** 04//07/2021

4 #P-Completeness of Common Problems in NP

4.1 SAT Variants

It is possible to modify the proof of the Cook-Levin Theorem to show that #SAT is #P-complete. From this starting point, we can produce several parsimonious reductions to show that other counting versions of common problems are #P-complete.

Theorem 1

- *There is a parsimonious reduction from SAT to 3SAT. Therefore, #3SAT is #P-complete.*
- *There is a parsimonious reduction from 3SAT to 3SAT-3. Therefore, #3SAT-3 is #P-complete.*
- *There is a parsimonious reduction from 3SAT-3 to CLIQ. Therefore, #CLIQ is #P-complete.*

The proof that #3SAT is #P-complete involves creating dummy variables with clauses forcing them to be false, in order to fill up all 1-clauses and 2-clauses while retaining the number of possible solutions. It then uses additional dummy variables to recursively deconstruct clauses of size ≥ 4 into smaller clauses which also retain the same number of possible solutions. This reduction is therefore parsimonious.

From there, the standard reductions from 3SAT to 3SAT-3, and from 3SAT-3 to CLIQ, are both parsimonious. Therefore, the counting versions of those problems are also #P-complete.

4.2 Planar SAT Variants

We can extend the chain of parsimonious reductions to Planar 3-SAT - the standard reduction from 3SAT to PL-3SAT is also parsmonious. This gets us that #PL-3SAT is #P-complete, and from there, we can prove #P-completeness of other planar satisfiability problems, such as #PL-RECT-3SAT.

However, in order to prove that #PL-1-in-3-SAT is #P-complete, we need a different reduction. The established reduction between 3SAT and PL-1-in-3-SAT is not c-monious, because most of the 3SAT solutions have one corresponding PL-1-in-3-SAT solution, but one particular case admits two PL-1-in-3-SAT solutions. There is, however, a parsimonious reduction from PL-RECT-3SAT to PL-POS-RECT-1-in-3-SAT, allowing us to conclude this.

4.3 Planar Directed Hamiltonian Cycles and Slitherlink

The standard reduction from Planar 3SAT to Planar Directed Hamiltonian Cycle is not parsimonious because we can create a distinct Hamiltonian cycle for each solution to a clause, meaning one clause may admit anywhere between one

and three solutions. However, an alternative reduction exists which is parsimonious. Sato (2002) constructed an XOR gadget which ensures that exactly one edge it connects is used in the Hamiltonian cycle, which allows for a parsimonious reduction from PL-1-in-3-SAT to PL-X3C, which is HAM CYCLE restricted to directed planar graphs of max degree 3. This reduction relies on three new gadgets (a binary OR gate, a ternary OR gate, and an implication gate) which all admit unique solutions.

From this, it is trivial to extend to the general problem of HAM CYCLE, and consequently prove that #HAM CYCLE is #P-complete.

Armed with this new problem, we may turn to the game of Slitherlink. This is known to be NP-complete by a reduction from grid-restricted Hamiltonian cycle, but the counting version of this Hamiltonian cycle variant is not yet known to be #P-complete. An alternative reduction is possible from PL-X3C which forces a unique solution for the cycle corresponding to each puzzle solution, and thus we get that #SLITHERLINK is #P-complete.

4.4 Permanent of a Matrix

Valiant proved all of these results about the permanent of a matrix [**?**].

Definition 6 *For* $M = (m_{i,j})$ *an* $n \times n$ *matrix, the* permanent *of* M*, denoted* $PERM(M)$*, is given by*

$$PERM(M) = \sum_{\pi} \prod_{i=1}^{n} m_{i,\pi(i)}$$

which is the sum of all possible products of exactly one element from each row and column.

The permanent differs from the determinant in that, when taking the determinant, we multiply each product by -1 before adding it if the number of inversions in its index permutation π is odd, whereas in the permanent, we do not do this. This unfortunately makes the permanent more challenging to compute, since Gaussian elimination can no longer be used - whereas the determinant can be computed in $O(n^3)$ time, computing the permanent is #P-complete. It was this drastic leap in complexity which led Valiant to further analyze the problem and consequently originally define the class #P to encapsulate its hardness.

In Valiant's work, he redefines the permanent of a matrix M of the sum of weights of all *cycle covers* of the weighted DAG whose adjacency matrix is M. A cycle cover is a set of cycles, not necessarily disjoint, such that every vertex is present in at least one cycle. The weight of a cycle here denotes the *product* (rather than sum) of the weights of its edges, while the weight of the entire cycle cover is the sum of each cycle's individual weight. Redefining the permanent as the sum of weights of cycle covers allowed Valiant to show that PERM is in #P.

The problem of computing the permanent is #P-complete. It remains so under various restrictions: it is #P-complete when restricted to matrix elements in $\{-1, 0, 1, 2, 3\}$, as well as when restricted to 0-1 matrices.

The problem PERMMOD of computing the permanent mod r for some parameter r is #P-hard but not known to be in #P. It can be shown that PERM $\leq_{p,o}$ PERMMOD. Since the permanent of a 0-1 matrix is bounded by $n!$, one may compute the least prime p for which the product of all primes $\leq p$ exceeds $n!$, call PERMMOD(M, r) on each of those primes, and then use the Chinese Remainder Theorem. Note that for fixed r this is polynomial time (it can be done in $O(n^{4r-3})$ operations) so computing the permanent with an upper bound on the matrix entries takes polynomial time.

4.5 Bipartite Matchings

Consider a graph G. A *matching* on G is a subset of edges such that no vertex is incident to more than one of the edges. The set MAT contains all graphs that have a matching. There are also special types of matchings: a *perfect matching* (PM) is one where every vertex is incident to some edge in the graph, and a *maximal matching* (MM) is one which cannot be extended into a larger valid matching by adding another edge from G. The counting problems of determining how many matchings of a given type G has (#MAT, #PM, and #MM) are all #P-complete. The versions of these problems restricted to bipartite graphs (#BIP-MAT, #BIP-PM, and #BIP-MM) are also all #P-complete.

By viewing a M as the adjacency matrix of a bipartite graph, we get that the permanent of the matrix is equal to the number of perfect matchings on the graph. This gives rise to a reduction showing that PERM $\leq_{p,o}$ #BIP-PM. From here, we can show that #BIP-PM $\leq_{p,o}$ #BIP-MAT and #BIP-PM $\leq_{p,o}$ #BIP-MM, via a pair of similar reductions that involve replacing each node with n^2 nodes and then replacing each edge with a copy of $K(n^2, n^2)$ between the two clusters. It follows from this that the general problems #MAT and #MM are thus also #P-complete.

4.6 #2SAT and Graph Problems

While the standard version of 2SAT is in P, the counting version #2SAT is #P-complete. However, one variant of 2SAT is more useful for us in proving hardness results, known as TH-POS-2SAT.

Definition 7 *The problem of* Threshold Positive 2SAT *(TH-POS-2SAT) is as follows: Given φ, a Boolean formula in 2CNF form with all positive literals, and a* threshold $t \in \mathbb{N}$*, determine whether there is a satisfying assignment for φ that has at least t variables assigned to False.*

The counting version of this problem, #TH-POS-2SAT, is #P-complete via a parsimonious reduction from PM to TH-POS-2SAT. The reduction takes a graph of order $2k$, generates a variable for each edge, creates a formula consisting of a 2-clause for each pair of incident edges, and then queries TH-POS-2SAT on this formula with threshold k. The variables assigned to False in a particular

matching correspond to edges we are including. No perfect matching has two incident edges, so no clause will be false in a perfect matching.

From TH-POS-2SAT, we are able to prove that #IND SET is #P-complete, via a similar parsimonious reduction. From a 2CNF all positive formula with threshold $\mathtt{k}$, we may build a graph with vertices for each variable and edges corresponding to each clause, and then search for an independent set of size $\mathtt{k}$. As a corollary, this is an alternative proof that #CLIQ is #P-complete, because IND SET and CLIQ are dual problems.

4.7 Another Solution Problems

This class of problems is a weaker way of measuring solution counts - it only asks us to determine whether a provided solution is unique.

Definition 8 *Consider a set* $\mathtt{A} \in NP$*. Then there exists a polynomial* $\mathtt{p}$ *and a set* $\mathtt{B} \in P$ *such that*

$$\mathtt{A} = \{\mathtt{x} : (\exists \mathtt{y})[|\mathtt{y}| = \mathtt{p}(|\mathtt{x}|) \wedge (\mathtt{x}, \mathtt{y}) \in \mathtt{B}]\}.$$

The Another Solution Problem *of* $\mathtt{A}$*, denoted* ASPA*, is the following: Given a problem instance* $\mathtt{x}$ *and a valid solution* $\mathtt{y}$*, determine if a different solution exists for* $\mathtt{x}$*. (You do not need to find the solution.)*

Sometimes, finding a solution to $\mathtt{A}$ is hard, but if you're given a solution it may be easy to show another exists and solve ASPA. For instance, ASP3COL is easy: take the given 3-coloring and permute the colors.

For another example, $\text{HAMC}_{\text{CUBIC}}$ is the NP-complete problem of finding a Hamiltonian cycle on a cubic graph. Tutte, Thomason, and Krawczyk proved three independent results that, when combined, confirm that given a Hamiltonian cycle on a cubic graph, a different one may be generated in worst case exponential time [**?**, **?**, **?**]. The problem $\text{ASPHAMC}_{\text{CUBIC}}$ is then solvable in $\mathcal{O}(1)$ time: just output "yes" every time! We do not care about finding the solution, only that another one exists, which will always be the case by these three results.

We can determine if a given ASP-version of a problem is hard via an *ASP-reduction*.

Definition 9 *Let* $\mathtt{A}, \mathtt{B} \in NP$*.*

1. $\mathtt{A}$ *is* ASP-reducible *to* $\mathtt{B}$ *if there is a parsimonious reduction from* $\mathtt{A}$ *to* $\mathtt{B}$ *which, given a solution for an instance of* $\mathtt{A}$*, allows a solution for an instance of* $\mathtt{B}$ *to be computed in polynomial time.*

2. $\mathtt{B}$ *is* ASP-complete *if, for all* $\mathtt{A} \in NP$*,* $\mathtt{A}$ *is ASP-reducible to* $\mathtt{B}$*.*

If $\mathtt{A}$ is ASP-reducible to $\mathtt{B}$, then we gain some insight into the relative hardness of ASP-versions of certain problems.

Theorem 2 *Suppose* A *is ASP-reducible to* B*. Then:*

- $\mathsf{ASPB} \in \mathsf{P} \Rightarrow \mathsf{ASPA} \in \mathsf{P}$.
- ASPA *is NP-hard* $\Rightarrow \mathsf{ASPB}$ *is NP-hard.*

The proof of this theorem is left as an exercise.

5 Open Problems

1. Is there a polynomial-time algorithm for solving some #P-complete problem? (Finding one implies P = NP. Proving none exists gives P $\subseteq$ NP $\subseteq$ #P, where at least one of the $\subseteq$ is strict.)
2. Similarly to the above, does #P = FP? (FP is the class of all function problems solvable by a deterministic Turing machine in polynomial time.) If the answer is yes, then P = NP = PH (where PH is the union of the entire polynomial hierarchy).
3. There is a set of classes that encapsulates a counting version of the W[t]-hierarchy - a parameterized counting problem is in #W[t] if it is fixed-parameter parsimonious reducible to #WSAT[t], the counting version of weighted circuit SAT of weft t. It is open whether the problem p-#MATCHING (the problem of counting the number of size-k matchings in a bipartite graph G for some input k) is in #W[1] [**?**].
4. It is open whether the entire W[t]-hierarchy is contained within #W[1].

6 Important Problems

- #SAT - #P-complete [**?**]

  Given a Boolean formula, find the number of satisfying assigments it has.

- #3SAT - #P-complete [**?**, **?**]

  Given a Boolean formula with exactly three literals per clause, find the number of satisfying assignments it has.

- #HAM CYCLE - #P-complete [**?**]

  Given an undirected graph, compute the number of distinct Hamiltonian cycles it has.

- PERMANENT - #P-complete [**?**]

  Given an $n \times n$ matrix M, compute $\sum_{\pi} \prod_{i=1}^{n} m_{i,\pi(i)}$.

- #MAT - #P-complete [**?**, **?**]

  Given a bipartite graph, determine the number of matchings it has. (Remains #P-complete even for perfect and maximal matchings, and even on bipartite graphs in any form.)

References

[1] Flum, Jörg, and Grohe, Martin. The parameterized complexity of counting problems. SIAM Journal on Computing, 33(4):892-922, 2012. https://doi.org/10.1137/S0097539703427203

[2] Adam Krawczyk. The complexity of finding a second hamiltonian cycle in cubic graphs.Journal of Computing and System Science, 58(3):641–647,1999.https://doi.org/10.1006/jcss.1998.1611

[3] Seta Takahiro. The complexities of puzzles, cross sum, and their another solution problems (senior honors thesis, university of tokyo). https://www.cs.umd.edu/users/gasarch/BLOGPAPERS/takahiro.pdf, 2002.

[4] Andrew B. Thomason. Hamiltonian cycles and uniquely edge colourable graphs. Annals of Discrete Mathematics, 3:259–268, 1978.

[5] Toda, Seinosuke. PP is as hard as the polynomial-time hierarchy. SIAM Journal on Computing, 20(5):865–877, 1991. https://doi.org/10.1137/0220053.

[6] William T Tutte. On Hamiltonian circuits. The Journal of the London Mathematical Society, pages 98–101, 1946.

[7] Valiant, Leslie. The complexity of computing the permanent. Theoretical Computer Science, 8:189–201, 1979. http://dx.doi.org/10.1016/0304-3975(79)90044-6