Counting Problems

➤ Examples of counting problems
➤ The class #P
➤ Reductions and completeness
➤ The class ⊕P

(C. Papadimitriou: Computational Complexity, Chapter 18)

Counting Problems—cont’d

➤ Counting the number of solutions can be highly nontrivial even if the decision problem is polynomial.
➤ An example is the problem of counting the number of perfect matchings of a bipartite graph.
➤ This corresponds to the problem of computing the permanent of a matrix

\[ \text{perm} A^G = \sum_{\pi} \prod_{i=1}^{n} A^G_{\pi(i)} \]

where \( A^G \) is the adjacency matrix of the graph.
➤ This is why the problem is often called PERMANENT.

Counting problems—cont’d

➤ Previously we have considered two types of problems: decision problems (whether a solution exists) and function (search) problems (find a solution).
➤ Now we consider a new type of a counting problem asking how many solutions exist.
➤ #SAT: given a Boolean expression, compute the number of different truth assignments that satisfy it.
➤ #HAMILTON PATH: compute the number of different Hamilton paths in a given graph.
➤ These are counting versions of NP-complete decision problems.

A bipartite graph with \( n \) “boys” \( \{u_1, \ldots, u_n\} \) and \( n \) “girls” \( \{v_1, \ldots, v_n\} \) can equivalently be seen as a directed graph with nodes \( \{1, \ldots, n\} \) where \( (i, j) \) is an edge in \( G' \) iff \( [u_i, v_j] \) is an edge in \( G \).

➤ Now a perfect matching corresponds to a cycle cover: a set of node-disjoint cycles that together cover all the nodes.

Example. [Papadimitriou, 1994]

For instance, a perfect matching \( \{[u_1, v_3], [u_3, v_2], [u_2, v_1], [u_4, v_4]\} \) corresponds to a cycle cover \( \{(1,3,2,1),(4,4)\} \).
Counting problems—cont’d

➤ Counting solutions is relevant, e.g., to probabilistic calculations.
➤ GRAPH RELIABILITY: count the number of subgraphs of a graph that contain a path from 1 to \( n \).

This number (divided by the number of subgraphs) gives the reliability of the graph: the probability that two nodes remain connected if all edges fail independently with probability \( \frac{1}{2} \).

The class \#P

➤ Let \( Q \) be a polynomially balanced and polynomial-time decidable binary relation. The counting problem associated with \( Q \) is the following: Given \( x \), how many \( y \) are there such that \((x, y) \in Q\) (the answer given as a binary integer).

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

➤ For \#SAT relation \( Q: (x, y) \in Q \) iff a truth assignment \( y \) satisfies a Boolean expression \( x \).
➤ For \#HAMILTON PATH relation \( Q: (x, y) \in Q \) iff \( y \) is a Hamilton path of a graph \( x \).

#P-Completeness

➤ Counting problems can be ordered using parsimonious reductions.
➤ A parsimonious reduction from a counting problem \( A \) to a counting problem \( B \) is a function \( R \) which maps an instance \( x \) of \( A \) to an instance \( R(x) \) of \( B \) such that the number of solutions of \( R(x) \) is the same as that of \( x \).
➤ Most reductions between \textbf{NP}-complete problems presented previously are parsimonious.
➤ A counting problem in \#P is \#P-complete if every problem in \#P can be reduced to it with a parsimonious reduction.

The class \#P—cont’d

Theorem. \#SAT is \#P-complete

Proof. Given \( A \in \#P \) with relation \( Q \) there is a poly-time TM \( M \) deciding \( Q \). We can build a circuit \( C(x) \) with \( |x|^k \) inputs s.t. with input \( y \) output of \( C(x) \) is true iff \( M \) accepts \( x; y \) (Cook’s theorem).

This is a parsimonious reduction to \#CIRCUIT SAT which reduces to \#SAT parsimoniously. (Parsimonious reductions compose.) \( \square \)

➤ This implies directly that many counting versions of \textbf{NP}-complete problems are \#P-complete.
➤ \#HAMILTON PATH is \#P-complete.
The class \#P—cont’d

➤ Note: a polynomial algorithm for a search problem does not imply that the corresponding counting problem is solvable in polynomial time.
➤ A classical example is PERMANENT
➤ The corresponding search problem (finding a perfect matching of a bipartite graph) is solvable in polynomial time.
➤ However, PERMANENT is \#P-complete.
➤ Notice that this implies that, for example, \#SAT can be reduced to PERMANENT with a parsimonious reduction. (Hence, the reduction has to be complicated and indirect!)

The class \#P—cont’d

➤ Notice that \#P problems can be solved in polynomial space.
➤ How do PH and \#P relate? (Remember: PH ⊆ PSPACE).
➤ Counting is stronger than the polynomial hierarchy!
➤ Toda’s theorem: PH ⊆ PP

where PP effectively tells only whether the first bit of the number of accepting computations is zero or one.

The class \⊕P

➤ What about deciding the last bit of the number of accepting computations?
➤ ⊕SAT: Given a set of clauses, is the number of satisfying truth assignments odd?
➤ \(L \in \oplus P\) if there is a nondeterministic Turing machine \(N\) such that for all strings \(x, x \in L\) iff the number of accepting computations of \(N\) on \(x\) is odd (or equivalently)
➤ \(L \in \oplus P\) if there is a polynomially balanced and polynomially decidable relation \(R\) such that \(x \in L\) iff the number of \(y\)s such that \((x, y) \in R\) is odd.

Theorem. \⊕P-cont’d

\(\oplus SAT\) and \(\oplus HAMILTON PATH\) are \(\oplus P\)-complete.

Theorem. \(\oplus P\) is closed under complement.

Proof. The complement of \(\oplus SAT\) (deciding whether the number of satisfying assignments is even) is \(co\oplus P\)-complete. We show that this problem reduces to \(\oplus SAT\) making \(\oplus SAT\) \(co\oplus P\)-complete. As \(\oplus SAT\) is also \(\oplus P\)-complete, \(\oplus P = co\oplus P\) (the classes are closed under reductions).

Reducing the complement of \(\oplus SAT\) to \(\oplus SAT\): Given a set of clauses on variables \(x_1, \ldots, x_n\), (i) add the new variable \(z\) to each clause and (ii) add \(n\) clause \(\neg z \lor x_i, i = 1, \ldots, n\). Now the number of satisfying truth assignment has increased by one, in which each variable true. □
⊕P—cont’d

➤ ⊕P seems weaker than PP: ⊕MATCHING is in P.
➤ But not powerless:

**Theorem.** NP ⊆ RP⊕P

Proof sketch.

➤ The idea is to show how an NP-complete problem (SAT) can be solved using a Monte Carlo algorithm which uses ⊕SAT as its oracle.

➤ For the algorithm we define for a set of Boolean variables \( S \subseteq \{x_1, \ldots, x_n\} \) a Boolean expression \( \eta_S \) stating that an even number among the variables in \( S \) are true as follows: Let \( y_0, \ldots, y_n \) be new variables. Now \( \eta_S \) is the conjunction of the expressions \( (y_0), (y_n) \), and for all \( i = 1, \ldots, n \), \((y_i \leftrightarrow (y_{i-1} \oplus x_i)), \) if \( x_i \in S \) and \((y_i \leftrightarrow y_{i-1}), \) if \( x_i \notin S \).

Proof—cont’d

➤ Clearly, the algorithm does not have any false positives
➤ It can be shown that the probability of a false negative is no larger than 7/8.
➤ Hence, by repeating the algorithm six times the probability of a false negative is less than half. ☐

Learning Objectives

➤ The concept of counting problems.
➤ Classes #P and ⊕P.
➤ Parsimonious reductions and completeness
➤ Typical complete problems for #P and ⊕P.
➤ The relationship of #P and ⊕P to other complexity classes.