Slide 1

NP-COMPLETE PROBLEMS

An alternative characterization of $\mathbf{NP}$:

- ➤ A relation $R \subseteq \Sigma^* \times \Sigma^*$ is polynomially decidable iff there is a deterministic TM deciding the language $\{x;y \mid (x,y) \in R\}$.
- ➤ R is polynomially balanced if $(x,y) \in R$ implies $|y| \leq |x|^k$ for some $k \geq 1$.

Proposition. Let $L \subseteq \Sigma^*$ be a language. $L \in \mathbf{NP}$ iff there is a polynomially balanced and polynomially decidable relation R such that $L = \{x \mid (x,y) \in R \text{ for some } y\}$.

Slide 2

Proof. ($\Leftarrow$) Suppose there is such a relation R. Then L is decided by a NTM that on input x, guesses a y of length at most $|x|^k$ and uses the machine for R to decide in polynomial time whether $(x,y) \in R$.

($\Rightarrow$) Suppose $L \in \mathbf{NP}$. Hence, there is a NTM N deciding L in time $|x|^k$ for some k.

Define R: $(x,y) \in R$ iff y is the encoding of an accepting computation of N on input x.

R polynomially balanced ($|y| \leq |x|^k$) and

polynomially decidable (since it can be checked in linear time whether y encodes an accepting computation of N on x).

As N decides L: $L = \{x \mid (x,y) \in R \text{ for some } y\}$. $\square$

Slide 3

- ➤ A problem is in $\mathbf{NP}$ if any "yes" instance x of the problem has at least one **succinct certificate** (or polynomial witness) y.
- ➤ $\mathbf{NP}$ contains a huge number of practically important, natural computational problems:
  - A typical computational problem is to construct a mathematical object satisfying certain specifications (path, a solution of equations, register allocation, routing, VLSI layout,...). This is the certificate.
  - The decision version of the problem is determine whether at least one such an object exists for the input.
  - The object is usually not very large compared to the input.
  - The specifications of the object are usually simple enough to be checkable in polynomial time.

Slide 4

- ➤ Most problems arising in computational practice are in $\mathbf{NP}$.
- ➤ Computational complexity theory: which problems in $\mathbf{NP}$ belong to $\mathbf{P}$ and which do not?
- ➤ $\mathbf{NP}$-completeness is the basic tool:

  Showing that a problem is $\mathbf{NP}$-complete implies that the problem is among the least likely to be in $\mathbf{P}$.

  (If an $\mathbf{NP}$-complete problem is in $\mathbf{P}$, then $\mathbf{NP} = \mathbf{P}$.)

Slide 5

$\mathbf{NP}$-completeness complements algorithm design techniques.

When a problem is shown $\mathbf{NP}$-complete efforts are usually directed to alternative approaches:

(i) Approximation algorithms
(ii) Attacking special cases
(iii) Studying average case performance
(iv) Randomized algorithms
(v) (Exponential) algorithms that are practical for small instances
(vi) Local search methods

Slide 6

Variants of Satisfiability

- Many problems if generalized enough become $\mathbf{NP}$-complete.
- Often it is important to find the dividing line between P and $\mathbf{NP}$-completeness.
- A basic technique is to investigate the subclass of the problem that is needed in the $\mathbf{NP}$-completeness proof to capture another $\mathbf{NP}$-complete problem.
- Next we consider the SAT problem as an example.

Slide 7

By kSAT we mean the subclass of SAT where all clauses have k literals.

Proposition. 3SAT is $\mathbf{NP}$-complete.

Proof.

- 3SAT is in $\mathbf{NP}$ as a special case of SAT which is in $\mathbf{NP}$.
- CIRCUIT SAT was shown to be $\mathbf{NP}$-complete and a reduction from CIRCUIT SAT to SAT has been presented.
- Consider now the clauses in the reduction. They have all at most 3 literals. Each clause with one or two literals can be modified to an equivalent clause with exactly 3 literals by duplicating a literal once or twice.
- Hence, we can reduce CIRCUIT SAT to 3SAT implying that 3SAT is $\mathbf{NP}$-complete. □

Slide 8

The subclass can sometimes be narrowed down by transformations:
Proposition. The subclass of SAT where each clause has at most 3 literals is $\mathbf{NP}$-complete even for expressions in which each variable is restricted to appear at most three times, and each literal at most twice.

- This is shown by a reduction where any instance of 3SAT is rewritten to eliminate the forbidden features.
- Consider a variable x appearing $k > 3$ times in the expression ϕ.
  (i) Replace the first appearance of x by x_1, the second by x_2, and so on where $x_1, \ldots, x_k$ are new variables.
  (ii) Add clauses $(\neg x_1 \lor x_2), (\neg x_2 \lor x_3), \ldots, (\neg x_k \lor x_1)$.
- Now ϕ is satisfiable iff the modified expression ϕ' is satisfiable as for each x appearing $k > 3$ times in ϕ, the truth values of $x_1, \ldots, x_k$ are the same for each truth assignment.

Slide 9

- The boundary between $\mathbf{P}$ and $\mathbf{NP}$-completeness is between 2SAT and 3SAT.
- Polynomial algorithm for 2SAT using graph reachability.
- Let ϕ be an instance of 2SAT. Define a graph $G(\phi)$ as follows: The vertices of $G(\phi)$ are the variables of ϕ and their negations. There is an arc (α, β) iff there is a clause $\neg\alpha \vee \beta$ or $\beta \vee \neg\alpha$ in ϕ.

Theorem. A Boolean expression ϕ is unsatisfiable iff there is a variable x such that there are paths from x to $\neg x$ and from $\neg x$ to x in $G(\phi)$.

Slide 10

Corollary. 2SAT is in $\mathbf{NL}$ ($\subseteq \mathbf{P}$).

Proof. $\mathbf{NL}$ is closed under complement. Hence it is sufficient to show that 2SAT COMPLEMENT is in $\mathbf{NL}$.

The condition of the theorem can be tested in logarithmic space non-deterministically by guessing a variable x and paths from x to $\neg x$ and back. □

- MAX2SAT generalizes 2SAT by asking whether there is a truth assignments which satisfies at least K clauses for a given integer K (but not necessarily all clauses as in 2SAT).

Theorem. MAX2SAT is $\mathbf{NP}$-complete.

Slide 11

NAESAT is a subclass of 3SAT such that there is a truth assignment where in no clause all three literals equal in truth value.

Theorem. NAESAT is $\mathbf{NP}$-complete.

Proof.

- CIRCUIT SAT was shown to be $\mathbf{NP}$-complete and a reduction from CIRCUIT SAT to SAT has been presented.
- Consider now the clauses in the reduction and add to all one- and two-literal clauses among them the same literal, say z, to make the clauses to have 3 literals.
- Claim: the resulting set of clauses is in NAESAT iff the original circuit is satisfiable.

Slide 12

($\Leftarrow$) Suppose that a truth assignment T satisfies all clauses (in the sense of NAESAT). Then the complementary truth assignment $\overline{T}$ satisfies the clauses in the sense of NAESAT, too.
In one of these z is $\mathbf{false}$ and, hence, the original set of clauses satisfiable implying that the circuit is satisfiable, as well.

($\Rightarrow$) Suppose a truth assignment satisfying the circuit.
Then there is a truth assignment T satisfying the clauses in the original reduction to SAT. Take $T(z) = \mathbf{false}$. Now in no clause all literals are $\mathbf{true}$ (they cannot be all $\mathbf{false}$): (i) clauses for $\mathbf{true}/\mathbf{false}$/NOT/variable gates contain z which is $\mathbf{false}$. (ii) For AND gates the clauses are: $(\neg g \vee h \vee z), (\neg g \vee h' \vee z), (g \vee \neg h \vee \neg h')$ where in the first two z is $\mathbf{false}$, and in the third all three cannot be true as then the first two would be not true. (iii) OR gates similarly. □

Slide 13

Graph-Theoretic Problems

- Consider undirected graphs.
- INDEPENDENT SET:
  INSTANCE: An undirected graph $G = (V, E)$ and an integer K.
  QUESTION: Is there an independent set $I \subseteq V$ with $|I| = K$.
  (A set $I \subseteq V$ is independent if $i, j \in I$ implies that there is no edge between i and j).

Theorem. INDEPENDENT SET is $\mathbf{NP}$-complete.

Slide 14

By examining the subclass of graphs needed in the reduction from SAT to INDEPENDENT SET:
Corollary. 4-DEGREE INDEPENDENT SET is $\mathbf{NP}$-complete.

- General pattern: problems in graph theory closely related.

CLIQUE:
INSTANCE: An undirected graph $G = (V, E)$ and an integer K.
QUESTION: Is there a clique $C \subseteq V$ with $|C| = K$.
(A set $C \subseteq V$ is clique if $i, j \in I$ implies that there is an edge between i and j).

NODE COVER:
INSTANCE: An undirected graph $G = (V, E)$ and an integer B.
QUESTION: Is there a set $C \subseteq V$ with $|C| \leq B$ such that if $[i, j] \in E$, then $i \in C$ or $j \in C$.

Slide 15

- INDEPENDENT SET can be reduced to CLIQUE: an independent set of a graph is a clique of the complement graph.
  Hence, G, K is reduced to $\overline{G}, K$.
- INDEPENDENT SET can be reduced to NODE COVER: I is an independent set of a graph iff $V - I$ is a node cover the graph.
  Hence, G, K is reduced to $G, |V| - K$.

Corollary. CLIQUE and NODE COVER are $\mathbf{NP}$-complete.

Slide 16

- A cut in an undirected graph $G = (V, E)$ is a partition of the nodes into two nonempty sets S and $V - S$. The size of a cut $(S, V - S)$ is the number of edges between S and $V - S$.
- MIN CUT (the problem of finding a cut with the smallest size) is in $\mathbf{P}$:
  (i) The smallest cut that separates two given nodes s and t equals to the maximum flow from s to t.
  (ii) Minimum cut: find the maximum flow between a fixed s and all other nodes and choose the smallest value found.
- But maximizing is harder:

Theorem. MAX CUT is $\mathbf{NP}$-complete.

Slide 17

- In application of graph partitioning, the sizes of S and $V - S$ are often constrained.
  MAX BISECTION is the problem of determining whether there is a cut $(S, V - S)$ with size of K or more such that $|S| = |V - S|$).
- Is MAX BISECTION easier than MAX CUT?

Lemma. MAX BISECTION is $\mathbf{NP}$-complete.

Proof. Reducing MAX CUT to MAX BISECTION by modifying input:

Add $|V|$ disconnected new nodes to G. Now every cut of G can be made a bisection by appropriately splitting the new nodes.

Now a graph $G = (V, E)$ has a cut $(S, V - S)$ with size of K or more iff the modified network has a cut with size of K or more such that $|S| = |V - S|$). □

Slide 18

- Minimization problem (MIN CUT + bisection requirement) is $\mathbf{NP}$-complete.
- BISECTION WIDTH: is there a bisection of size K or less?

Theorem. BISECTION WIDTH is $\mathbf{NP}$-complete.

Proof. A reduction from MAX BISECTION:

a graph $G = (V, E)$ where $|V| = 2n$ for some n has a bisection of size K or more iff the complement of G has a bisection of size $n^2 - K$ or less. □

Slide 19

Theorem. HAMILTON PATH is $\mathbf{NP}$-complete.

Corollary. TSP(D) is $\mathbf{NP}$-complete.

Proof. A reduction from HAMILTON PATH to TSP(D).

Given a graph G with n nodes, construct a distance matrix d_{ij} and a budget B such that there is a tour of length B of less iff G has a Hamilton path.

There are n cities and the distance $d_{ij} = 1$ if there is an edge $[i, j]$ in G and 2 otherwise. Finally $B = n + 1$. □

Slide 20

Sets and numbers

TRIPARTITE MATCHING:
INSTANCE: Sets B, G, H each containing n elements and a ternary relation $T \subseteq B \times G \times H$
QUESTION: Is there a set of n triplets in T no two of which have a component in common.

Theorem. TRIPARTITE MATCHING is $\mathbf{NP}$-complete.

Slide 21

Other set problems:

- SET COVERING:
  INSTANCE: A family $F = \{S_1, \ldots, S_n\}$ of subsets of a finite set U and a integer B.
  QUESTION: Is there a set of B sets in F whose union is U.
- SET PACKING:
  INSTANCE: A family $F = \{S_1, \ldots, S_n\}$ of subsets of a finite set U and a integer K.
  QUESTION: Is there a set of K pairwise disjoint sets in F.
- EXACT COVER BY 3-SETS:
  INSTANCE: A family $F = \{S_1, \ldots, S_n\}$ of subsets of a finite set U such that $|U| = 3m$ for some integer m and for all i $|S_i| = 3$.
  QUESTION: Is there a set of m sets in F that are disjoint and have U as their union.

Slide 22

Corollary. SET COVERING, SET PACKING, and EXACT COVER BY 3-SETS are $\mathbf{NP}$-complete.
Proof. (By generalization)

- TRIPARTITE MATCHING can be reduced to EXACT COVER BY 3-SETS by noticing that it is a special case where U is partitioned in 3 equal sets B, G, H and $F = \{\{b, g, h\} \mid b \in B, g \in G, h \in H\}$.
- EXACT COVER BY 3-SETS can be reduced to SET COVERING as a special case where the universe has $3m$ elements, all sets in F have 3 elements and the budget $B = m$.
- EXACT COVER BY 3-SETS can be reduced to SET PACKING as a special case where the universe has $3m$ elements, all sets in F have 3 elements and limit $K = m$. □

Slide 23

INTEGER PROGRAMMING:
INSTANCE: a system of linear inequalities with integer coefficients
QUESTION: Is there an integer solution of the system.

Corollary. INTEGER PROGRAMMING is $\mathbf{NP}$-complete.

Proof. SET COVERING reducible to INTEGER PROGRAMMING:
Given a family $F = \{S_1, \ldots, S_n\}$ of subsets of a finite set U and a integer B, construct a system (where $x_i = 1$ if S_i in the cover and otherwise $x_i = 0$):

$$\begin{array}{ll} 0 \leq x_1 \leq 1, \ldots, 0 \leq x_n \leq 1 & a_{11}x_1 + \cdots + a_{1n}x_n \geq 1 \\ \Sigma_{i=1}^{n} x_i \leq B & \vdots \\ & a_{m1}x_1 + \cdots + a_{mn}x_n \geq 1 \end{array}$$

where $a_{ij} = 1$ if ith element of U is in the set S_j, otherwise $a_{ij} = 0$. □

Slide 24

- LINEAR PROGRAMMING (INTEGER PROGRAMMING where non-integer solutions are allowed) is in $\mathbf{P}$.
- KNAPSACK:
  INSTANCE: A set of n items with each item i having a value v_i and a weight w_i both positive integers and integers W and K.
  QUESTION: Is there a subset S of the items such that $\Sigma_{i \in S} w_i \leq W$ but $\Sigma_{i \in S} v_i \geq K$.

Theorem. KNAPSACK is $\mathbf{NP}$-complete.

Slide 25

We show that a simple special case of KNAPSACK is $\mathbf{NP}$-complete:
INSTANCE: A set of integers $w_1, \ldots, w_n$ and an integer K.
QUESTION: Is there a subset S of the intergers such that $\Sigma_{i \in S} w_i = K$.

Proof.

- Reduction from EXACT COVER BY 3-SETS where $U = \{1, 2, \ldots, 3m\}$ and the sets S_i are given as bit vectors $\{0,1\}^{3m}$ and $K = 2^{3m} - 1$, i.e., find a subset of bit vectors that sum to K.
- This does not quite work because of the carry bit.
- A correction: use base $n+1$ instead of 2.
  S_i is the integer $w_i = \Sigma_{j \in S}(n+1)^{3m-j}$ and $K = \sum_{j=0}^{3m-1}(n+1)^j$

Slide 26

Pseudopolynomial algorithms and strong $\mathbf{NP}$-completeness

Proposition. Any instance of KNAPSACK can be solved in $\mathcal{O}(nW)$ time where n is the number of items and W is the weight limit.

Proof. Define $V(w,i)$: the largest value attainable be selecting some among the first i items so that their total weight is exactly w.

Each $V(w,i)$ ($w = 1, \ldots, W$, $i = 1, \ldots, n$) can be computed as follows:

$V(w, i+1) = \max\{V(w,i), v_{i+1} + V(w - w_{i+1}, i)\}$

where $V(w,0) = 0$ for all w.

For each entry this can be done in constant number of steps and there are nW entries.
An instance of KNAPSACK is answered “yes” iff there is an entry $V(w,i) \geq K$. $\square$

Slide 27

- Do we have $\mathbf{P} = \mathbf{NP}$ established now?
- The algorithm is not polynomial w.r.t. the length of the input (which $\mathcal{O}(n \log W)$) but exponential ($W = 2^{\log W}$).
- An algorithm where the time bound is polynomial in the integers in the input (not their logarithms) is called pseodopolynomial.
- A problem is called **strongly $\mathbf{NP}$-complete** if the problem remains $\mathbf{NP}$-complete even if any instance of length n is restricted to contain integers of size at most $p(n)$, a polynomial.
  ☞ Strongly $\mathbf{NP}$-complete problems cannot have pseodopolynomial algorithms (unless $\mathbf{P} = \mathbf{NP}$).
- Strongly $\mathbf{NP}$-complete problems:
  SAT, MAX CUT, TSP(D), HAMILTON PATH, ...
  Exception: KNAPSACK.

Slide 28

BIN PACKING:
INSTANCE: N positive integers $a_1, \ldots, a_N$ (items) and integers C (capacity) and B (number of bins).
QUESTION: Is there a partition of the numbers into B subsets such that for each subset S, $\Sigma_{a_i \in S} a_i \leq C$.

- BIN PACKING is strongly $\mathbf{NP}$-complete:
  Even if the integers are restricted to have polynomial values (w.r.t. the length of input), BIN PACKING remains $\mathbf{NP}$-complete.
- Any pseodopolynomial algorithm for BIN PACKING would yield a polynomial algorithm for all problems in $\mathbf{NP}$ implying $\mathbf{P} = \mathbf{NP}$.