NP-completeness Problems

Guan-Shieng Huang

Nov. 1, 2006

NP-completeness Problems

NP: the class of languages decided by nondeterministic Turing machine in polynomial time

NP-completeness:
Cook's theorem: SAT is NP-complete.

Certificate of TM:
Hard to find an answer if there is one, but easy to verify.

SAT — a satisfying truth assignment

HAMILTON PATH — a Hamilton path

Variants of Satisfiability

- k-SAT
- 3-SAT
- 2-SAT
- MAX 2SAT
- NAESAT

k-SAT: Each clause has at most k literals.
$(l_1 \vee l_2 \vee \cdots \vee l_t,\ t \leq k)$

Proposition 9.2 3-SAT is NP-complete.

For any clause $C = l_1 \vee l_2 \vee \cdots \vee l_t$, we introduce a new variable x and split C into

$$C_1 = l_1 \vee l_2 \vee \cdots \vee l_{t-2} \vee x,$$

$$C_2 = \neg x \vee l_{t-1} \vee l_t.$$

Each time we obtain a clause with 3 literals. Then $F \wedge C$ is satisfiable iff $F \wedge C_1 \wedge C_2$ is satisfiable

Knapsack: $\{1, 2, \ldots, n\}$, n items. Item i has value $v_i > 0$ and weight $w_i > 0$. Try to find a subset $S \subseteq \{1, \ldots, n\}$ such that $\sum_{i \in S} w_i \leq W$ and $\sum_{i \in S} v_i \geq K$ for some W and K.

Theorem 9.10 KNAPSACK is NP-complete.

Proposition 9.4 Any instance of KNAPSACK can be solved in $O(nW)$ time, where n is the number of items and W is the weight limit.

We can solve this by dynamic programming.
$V(w, i)$: the largest value attainable by selecting some among the i first items so that the total weight is exactly w.

$$\begin{cases} V(w, i+1) = \max\{V(w, i), v_{i+1} + V(w - w_{i+1}, i)\}; \\ V(w, 0) = 0. \end{cases}$$

If there is an entry $\geq K$, then answer "yes."
In fact, this is an example of pseudo-polynomial time algorithm.

Strong NP-completeness

Definition A decision problem is called strongly NP-complete iff it is still NP-complete even if any instance of length n is restricted to contain integers of size at most $p(n)$, a polynomial.

Theorem Strongly NP-complete problems have no pseudo-polynomial time algorithm, unless $P = NP$.