Section 23.3: NP: Problems with Easily Recognized Solutions

Definition of the Complexity Class NP

Big idea: efficient recognition of purported solutions.

Definition: A search problem belongs to the complexity class NP if and only if:

(1) Candidate solutions have length (in bits) polynomial in the input size.

(2) Can confirm or deny feasibility of a candidate solution in time polynomial in the input size.

Examples of Problems in NP

1. Search version of TSP
2. 3-SAT
3. Hamiltonian path
4. Search version of makespan minimization
5. Search version of independent set

etc.

NP Problems Are Solvable via Exhaustive Search

Generic Exhaustive Search for an NP Problem

n = size of input

c_1, c_2, d_1, d_2 constants (indep. of n)

(1) Enumerate candidate solutions, one by one.

[all have length (in bits) $\leq c_1 n^{d_1}$ => "only" $2^{c_1 n^{d_1}}$ to enumerate]

(2) Return first feasible solution found.
[or if none, report "no solution"]

[running time $\leq c_2 n^{d_2}$ per step of enumeration]

=> overall running time "only" exponential in n

NP-Hard Problems

Note: if every NP problem reduces to the TSP => very strong evidence of intractability (if any NP problem is hard, so is the TSP).

Definition: A problem is NP-hard if every problem in the class NP reduces to it. (solve the problem => solve every problem with efficiently recognizable solution)

↳ via a Cook reduction

Note: only search problems belong to NP, but any type of problem can be NP-hard (e.g., TSP).

Warning: more liberal definition than in many textbooks.

Proof Sketch of the Cook-Levin Theorem (I)

Cook-Levin Theorem: Every problem in NP reduces to 3-SAT.

Gist of the proof: Fix arbitrary NP problem A.

- description length $\leq c_1 n^{d_1}$ bits
- verification time $\leq c_2 n^{d_2}$

Ingredients:

- $c_1 n^{d_1}$ solution variables
  [to encode bits of a feasible solution of given instance of A]
- a $c_2 n^{d_2} \times c_2 n^{d_2}$ table of state variables
  (rows $\leftrightarrow$ time steps, columns $\leftrightarrow$ bits of memory)

Proof Sketch of the Cook-Levin Theorem (II)

- Solution variables = encode candidate solution
- State variables = encode verification algorithm's execution on a candidate solution

Constraints: enforce semantics of the state variables.

Ex: ith bit of memory at time t consistent with relevant memory contents at time t-1, given instance of problem A, values of solution variables, description of verification algorithm.

Intuition: not much happens in one time step => can enforce consistency with small # of disjunctions with ≤ 3 literals each.

finally: verification algorithm accepts candidate soln encoded by solution variables.