NP Completeness VI

2023/11/22

詹博华（中国科学院软件研究所）

Summary

In this lecture, we give more examples of NP-completeness proofs (exercises and problems in the textbook).

- Integer linear-programming.
- Independent set.
- Scheduling with profits and deadlines.

Review: linear-programming

- **Recall the linear-programming problem:**

$$\text{Find } \boldsymbol{x} \text{ that minimizes } \boldsymbol{c} \cdot \boldsymbol{x}$$

$$\text{subject to } \boldsymbol{a_i} \cdot \boldsymbol{x} \le b_i, \quad i = 1, \dots, m$$

here $\boldsymbol{c}, \boldsymbol{x}$, and each $\boldsymbol{a_i}$ are vectors.

- **The corresponding decision problem is:** determine whether

$$\boldsymbol{a_i} \cdot \boldsymbol{x} \le b_i, \quad i = 1, \dots, m$$

has a solution.

- In this class we studied the simplex method, which is usually efficient. Another method based on interior points have guaranteed polynomial time.

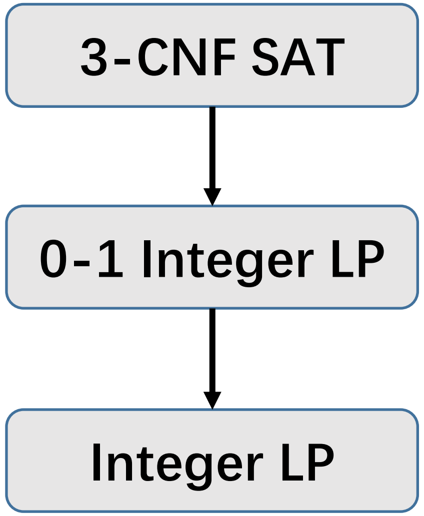
Integer linear-programming

- Similar setting as before, but entries in $\boldsymbol{x}$ are constrained to be integers.
- Given vectors $\boldsymbol{a_i}$ with integer entries, and integers b_i, determine whether there exists integer vector $\boldsymbol{x}$ such that

$$\boldsymbol{a_i} \cdot \boldsymbol{x} \leq b_i, \quad i = 1, \dots, m$$

Integer linear-programming is NP-complete

- We show by reduction from 3-CNF satisfiability, through an intermediate problem called 0-1 integer-programming.
- **0-1 integer-programming:** same setting as integer linear-programming, except entries in $\boldsymbol{x}$ are constrained to lie in set $\{0,1\}$.



Reduction from 3-CNF to 0-1 Integer LP (Exercise 34.5-2)

- **Goal:** given a 3-CNF formula $\phi = C_1 \land C_2 \land \cdots C_k$ over variables $x_1, x_2, \ldots, x_n$, construct a 0-1 integer programming problem that is solvable if and only if ϕ is satisfiable.
- Each boolean variable x_i in ϕ corresponds to an 0-1 variable x_i in 0-1 integer programming.

Reduction from 3-CNF to 0-1 Integer LP

- Each clause C_j can be translated into a constraint on the variables x_i, indicating that at least one of the literals is true.
- For example: if $C_j = x_1 \vee \neg x_2 \vee \neg x_3$, then the corresponding constraint is

$$x_1 + (1 - x_2) + (1 - x_3) \geq 1, \text{ or}$$

$$-x_1 + x_2 + x_3 \leq 1.$$

Reduction from 3-CNF to 0-1 Integer LP: Example

- Consider again the 3-CNF formula $\phi = C_1 \wedge C_2 \wedge C_3 \wedge C_4$, where

$$C_1 = (x_1 \vee \neg x_2 \vee \neg x_3),\ C_2 = (\neg x_1 \vee \neg x_2 \vee \neg x_3)$$
$$C_3 = (\neg x_1 \vee \neg x_2 \vee x_3),\ C_4 = (x_1 \vee x_2 \vee x_3)$$

- The corresponding 0-1 integer programming problem is:

$$x_1 + (1 - x_2) + (1 - x_3) \geq 1$$
$$(1 - x_1) + (1 - x_2) + (1 - x_3) \geq 1$$
$$(1 - x_1) + (1 - x_2) + x_3 \geq 1$$
$$x_1 + x_2 + x_3 \geq 1$$

- One solution is $x_1 = 1, x_2 = 1, x_3 = 0$.

Reduction from 0-1 Integer LP to Integer LP

- Next, we prove NP-completeness of integer linear-programming by reduction from 0-1 integer programming.
- Use the same variables and constraints, but also adding the constraint $0 \leq x_i \leq 1$ (that is, $-x_i \leq 0, x_i \leq 1$).
- For example, the problem on the previous slide becomes:

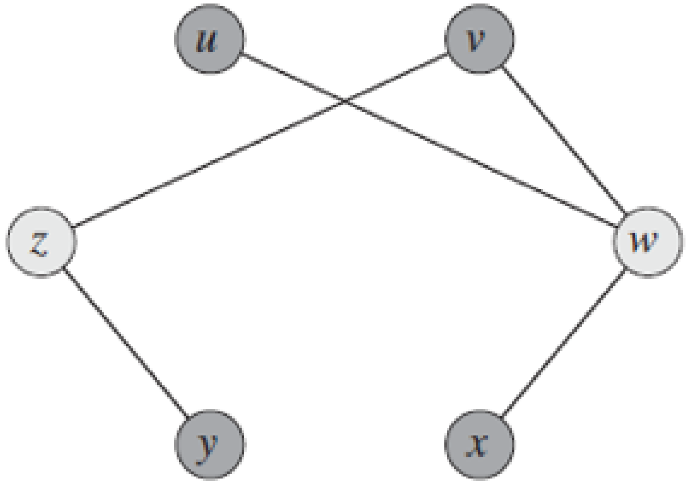
$$x_1 + (1 - x_2) + (1 - x_3) \geq 1$$
$$(1 - x_1) + (1 - x_2) + (1 - x_3) \geq 1$$
$$(1 - x_1) + (1 - x_2) + x_3 \geq 1$$
$$x_1 + x_2 + x_3 \geq 1$$
$$0 \leq x_1 \leq 1, \qquad 0 \leq x_2 \leq 1, \qquad 0 \leq x_3 \leq 1$$

Reduction from 0-1 Integer LP to Integer LP

- The integer LP problem with added constraints has a solution if and only if the original 0-1 integer LP problem has a solution.
- **Conclusion:** integer linear-programming is NP-complete.

Independent set

- We consider another problem on graphs. Given a graph $G = (V, E)$, a subset $V' \subseteq V$ is an independent set if each edge in E is incident on at most one vertex in V.
- **Example:** the graph below has independent set of size 4, given by $\{u, v, x, y\}$.

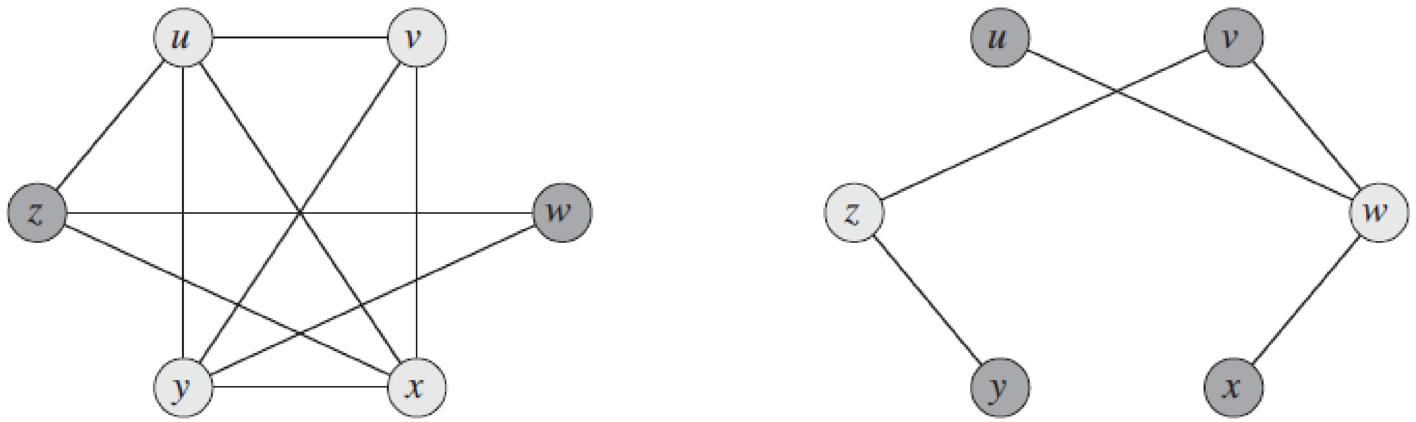


Independent set is NP-complete (34-1)

- **Decision problem:** given a graph G and integer k, determine whether there exists an independent set of size k.
- Proof by reduction from the clique problem (very similar to vertex cover).

Reduction from clique to independent set

- Given a clique problem with graph G (left), construct the complement graph G'.



- A subset of vertices V' is an independent set for G' if and only if V' is a clique for G.

Reduction from clique to independent set

- Hence, a graph G has a clique of size k if and only if graph G' has an independent set of size k.
- **Conclusion:** independent set problem is NP-complete.
- **Note:** this problem has polynomial solution if each vertex of G has degree 2 (then the graph consists of simple cycles and paths).
- The problem is also solvable in polynomial time if G is bipartite (use techniques from maximum-bipartite-matching).

Scheduling with profits and deadlines (34-4)

- Consider the following scheduling problem: we have one machine and a set of n tasks $a_1, a_2, \dots, a_n$. Each task requires time t_i, deadline d_i, and profit p_i.
- If task a_i is completed before time t_i, then profit p_i is received. Otherwise no profit is received.
- Find a schedule that maximizes the total profit.

Task scheduling with profits and deadlines is NP-complete

- **Decision problem:** given task information (t_i, d_i, p_i), and a target profit P, determine whether it is possible to earn at least profit P.
- We now show this problem is NP-complete, by reduction from the subset-sum problem.

Reduction from subset-sum to task scheduling

- Given a subset-set problem: a finite set of integers S and a target t, is there are subset $S' \subseteq S$ that sums to t?
- Construct the task scheduling problem as follows:
  - One task for each integer n_i in S, with time n_i, deadline t, and profit n_i.
  - The target profit is t.
- Clearly, the amount of profit earned equals the total time the machine is in used. To earn profit t, the machine must be continuously in use. That is, we must find a subset of tasks whose total time is exactly t.

Reduction from subset-sum to task scheduling

- Hence, it is possible to obtain a profit of t only if there exists a subset of numbers in S whose sum is t – that is, the original subset-sum problem is solvable.
- **Conclusion:** task scheduling with profits and deadlines is NP-complete.
- **Note:** this problem has a polynomial solution (by dynamic programming) if the processing times t_i are of size $O(n)$).

Conclusion

- **NP-complete problems are the hardest problems in class NP, and are widely believed to have no polynomial-time algorithms** (unless P = NP).
- If we demonstrate a problem is NP-complete, we can stop looking for general polynomial solutions, instead look for special cases, approximation algorithms (tomorrow), and heuristic algorithms.
- NP-completeness is proved by reduction from a problem that is already known to be NP-complete, and we already have many such starting points (3-CNF SAT, clique, subset-sum, etc).