NP Completeness V

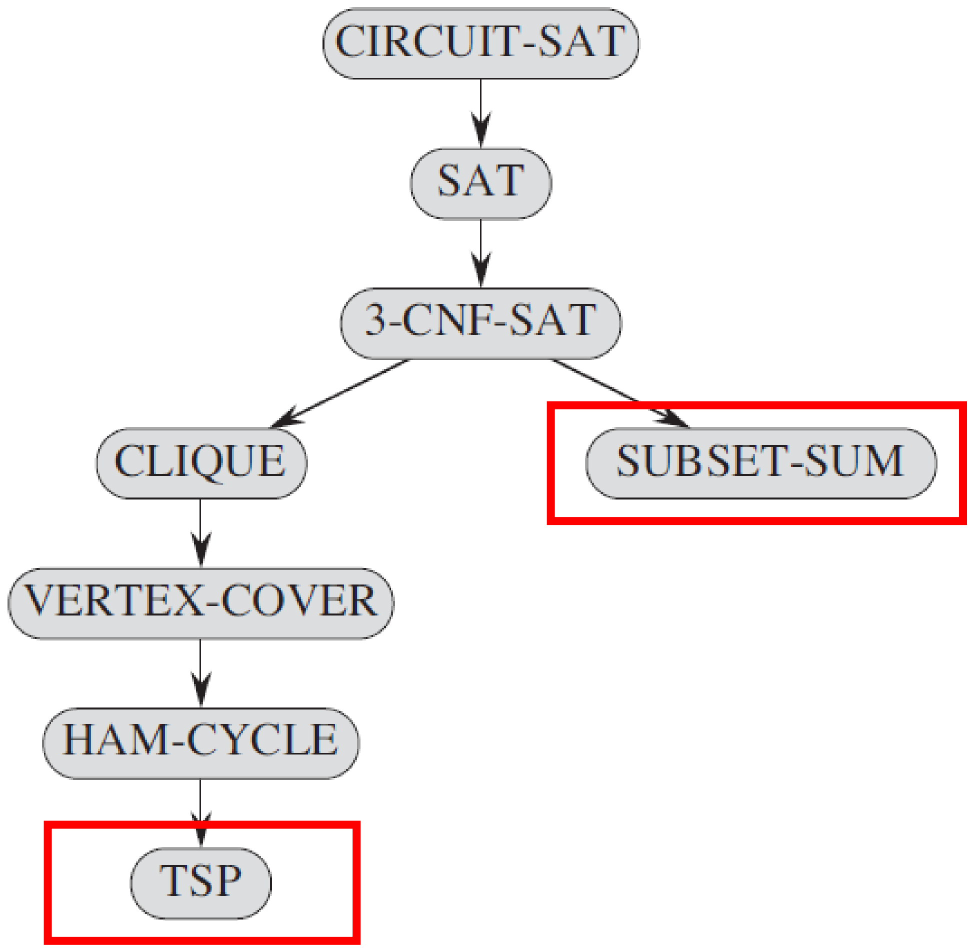
2023/11/22

詹博华（中国科学院软件研究所）

More NP-complete problems

In this lecture, we show:

- Traveling salesman problem is NP-complete, by reducing from Hamiltonian cycle.
- Subset-sum problem is NP-complete, by reducing from 3-CNF satisfiability.



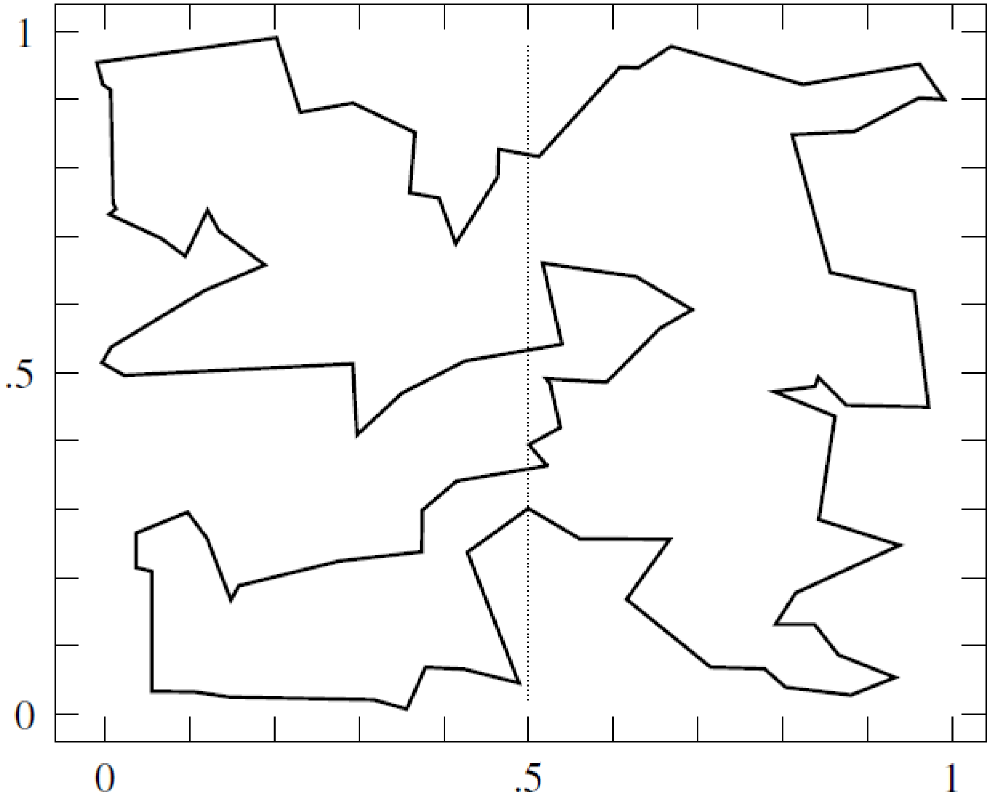
Traveling Salesman Problem

- Closely related to the Hamiltonian cycle problem.
- Suppose a salesman must visit n cities, where the cost to travel between each pair of cities is known. In which order should the salesman visit the cities so the total cost is minimized?
- **Optimization problem:** given a graph G, suppose there is an edge between each pair of vertices i and j, with cost $c(i, j)$. Find a Hamiltonian tour of G with minimal total cost.
- **Decision problem:** given graph G and integer k, does there exist a Hamiltonian cycle with total cost at most k?

Traveling Salesman Problem

- Traveling Salesman Problem looks similar to shortest path, but is well-known to (probably) have no polynomial solution.

- There are many approximation / heuristic techniques that work well for TSP in practice.
- Right: an example of TSP solved using simulated annealing. (Source: Numerical Recipes, The Art of Scientific Computing)

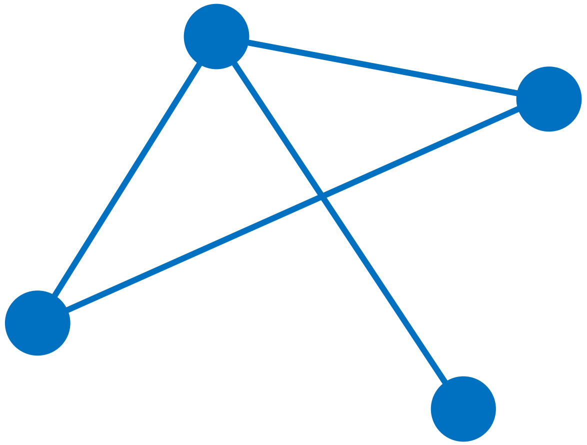


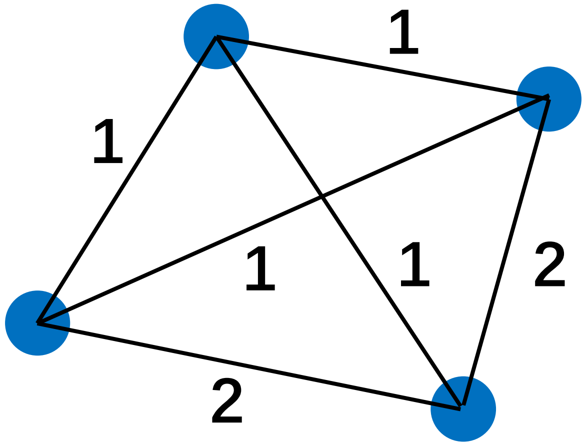
Reduction

- Given an instance of Hamiltonian cycle problem: a graph G and we wish to determine whether G has a Hamiltonian cycle.
- Construct G' as follows: the vertices of G' are the same as that for G. For each pair of vertices (i, j), let $c(i, j) = 1$ if there is an edge between i and j in G, and $c(i, j) = 2$ if otherwise.

Reduction

- G has a Hamiltonian cycle if and only if there is a tour in G' with total cost n (it is possible to visit all n cities by traveling only along edges of G).
- Example: the left graph has no Hamiltonian cycle, and the shortest cycle on the right has length 5.
- **Conclusion:** Traveling salesman problem is NP-complete.





Subset-sum problem

- Given a finite set S of positive integers, and a target $t > 0$, does there exist a subset $S' \subseteq S$ whose elements sum to t.
- For example, if

$$S = \{1, 2, 7, 14, 49, 98, 343, 686, 2409, 2793, 16808, 17206, 117705, 117993\}$$

and $t = 138457$, then

$$S' = \{1, 2, 7, 98, 343, 686, 2409, 17206, 117705\}$$

is a solution.

- Note the size of the problem is the total number of digits of elements in S, not sizes of numbers in S.

Idea of proof

- **Reduction from 3-CNF satisfiability.**
- Given a 3-CNF formula ϕ over variables $x_1, x_2, \ldots, x_n$, with clauses $C_1, C_2, \ldots, C_k$, each clause containing exactly three distinct literals, construct a subset-sum problem that can be solved if and only if ϕ is satisfiable.
- We use base 10 for expressing the numbers in the subset-sum problem. The base need to be sufficiently large to prevent carries from lower digits to higher digits.

Example of translation

For the formula $C_1 \wedge C_2 \wedge C_3 \wedge C_4$, where:

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

		x_1	x_2	x_3	C_1	C_2	C_3	C_4
v_1	=	1	0	0	1	0	0	1
v_1'	=	1	0	0	0	1	1	0
v_2	=	0	1	0	0	0	0	1
v_2'	=	0	1	0	1	1	1	0
v_3	=	0	0	1	0	0	1	1
v_3'	=	0	0	1	1	1	0	0
s_1	=	0	0	0	1	0	0	0
s_1'	=	0	0	0	2	0	0	0
s_2	=	0	0	0	0	1	0	0
s_2'	=	0	0	0	0	2	0	0
s_3	=	0	0	0	0	0	1	0
s_3'	=	0	0	0	0	0	2	0
s_4	=	0	0	0	0	0	0	1
s_4'	=	0	0	0	0	0	0	2
t	=	1	1	1	4	4	4	4

Step 1:

- Each number has $n + k$ digits, where n is the number of variables and k is the number of clauses.
- The highest n digits correspond to variables. The lowest k digits correspond to clauses.

		x_1	x_2	x_3	C_1	C_2	C_3	C_4
v_1	=	1	0	0	1	0	0	1
v_1'	=	1	0	0	0	1	1	0
v_2	=	0	1	0	0	0	0	1
v_2'	=	0	1	0	1	1	1	0
v_3	=	0	0	1	0	0	1	1
v_3'	=	0	0	1	1	1	0	0
s_1	=	0	0	0	1	0	0	0
s_1'	=	0	0	0	2	0	0	0
s_2	=	0	0	0	0	1	0	0
s_2'	=	0	0	0	0	2	0	0
s_3	=	0	0	0	0	0	1	0
s_3'	=	0	0	0	0	0	2	0
s_4	=	0	0	0	0	0	0	1
s_4'	=	0	0	0	0	0	0	2
t	=	1	1	1	4	4	4	4

Step 2:

- For each variable x_i, add two integers v_i and v_i', that have 1 on digit x_i and each C_j that contains x_i (resp. $\neg x_i$).
- For example, x_1 appears in C_1 and C_4, and $\neg x_1$ appears in C_2 and C_3.
- Similarly, x_2 appears in C_4, $\neg x_2$ appears in C_1, C_2, C_3. x_3 appears in C_3, C_4, $\neg x_3$ appears in C_1, C_2.

		x_1	x_2	x_3	C_1	C_2	C_3	C_4
v_1	=	1	0	0	1	0	0	1
v_1'	=	1	0	0	0	1	1	0
v_2	=	0	1	0	0	0	0	1
v_2'	=	0	1	0	1	1	1	0
v_3	=	0	0	1	0	0	1	1
v_3'	=	0	0	1	1	1	0	0
s_1	=	0	0	0	1	0	0	0
s_1'	=	0	0	0	2	0	0	0
s_2	=	0	0	0	0	1	0	0
s_2'	=	0	0	0	0	2	0	0
s_3	=	0	0	0	0	0	1	0
s_3'	=	0	0	0	0	0	2	0
s_4	=	0	0	0	0	0	0	1
s_4'	=	0	0	0	0	0	0	2
t	=	1	1	1	4	4	4	4

Step 3:

- For each clause C_j, add two rows s_j, s_j' (called *slack rows*).
- s_j has 1 at C_j and 0 everywhere else.
- s_j' has 2 at C_j and 0 everywhere else.

		x_1	x_2	x_3	C_1	C_2	C_3	C_4
v_1	=	1	0	0	1	0	0	1
v_1'	=	1	0	0	0	1	1	0
v_2	=	0	1	0	0	0	0	1
v_2'	=	0	1	0	1	1	1	0
v_3	=	0	0	1	0	0	1	1
v_3'	=	0	0	1	1	1	0	0
s_1	=	0	0	0	1	0	0	0
s_1'	=	0	0	0	2	0	0	0
s_2	=	0	0	0	0	1	0	0
s_2'	=	0	0	0	0	2	0	0
s_3	=	0	0	0	0	0	1	0
s_3'	=	0	0	0	0	0	2	0
s_4	=	0	0	0	0	0	0	1
s_4'	=	0	0	0	0	0	0	2
t	=	1	1	1	4	4	4	4

Step 4:

- The target t has 1 at each variable, and 4 at each clause.

		x_1	x_2	x_3	C_1	C_2	C_3	C_4
v_1	=	1	0	0	1	0	0	1
v_1'	=	1	0	0	0	1	1	0
v_2	=	0	1	0	0	0	0	1
v_2'	=	0	1	0	1	1	1	0
v_3	=	0	0	1	0	0	1	1
v_3'	=	0	0	1	1	1	0	0
s_1	=	0	0	0	1	0	0	0
s_1'	=	0	0	0	2	0	0	0
s_2	=	0	0	0	0	1	0	0
s_2'	=	0	0	0	0	2	0	0
s_3	=	0	0	0	0	0	1	0
s_3'	=	0	0	0	0	0	2	0
s_4	=	0	0	0	0	0	0	1
s_4'	=	0	0	0	0	0	0	2
t	=	1	1	1	4	4	4	4

Remainder of proof

- Since there is no possibility of carrying, we can consider the sum at each column independently.
- For the x_i columns to sum to 1, we need to choose exactly one of v_i and v_i', corresponding to setting x_i to true or false.
- Here we set $x_1 = \text{false}$, $x_2 = \text{false}$, $x_3 = \text{true}$.

		x_1	x_2	x_3	C_1	C_2	C_3	C_4
v_1	=	1	0	0	1	0	0	1
v_1'	=	1	0	0	0	1	1	0
v_2	=	0	1	0	0	0	0	1
v_2'	=	0	1	0	1	1	1	0
v_3	=	0	0	1	0	0	1	1
v_3'	=	0	0	1	1	1	0	0
s_1	=	0	0	0	1	0	0	0
s_1'	=	0	0	0	2	0	0	0
s_2	=	0	0	0	0	1	0	0
s_2'	=	0	0	0	0	2	0	0
s_3	=	0	0	0	0	0	1	0
s_3'	=	0	0	0	0	0	2	0
s_4	=	0	0	0	0	0	0	1
s_4'	=	0	0	0	0	0	0	2
t	=	1	1	1	4	4	4	4

Remainder of proof

- For each clause C_j, the contribution of rows v_i and v_i' to the sum at C_j is exactly the number of literals that are satisfied.
- If no literal in C_j is satisfied, then it is impossible for column C_j to sum to 4.
- Otherwise, can always add one or both of the slack rows s_j, s_j'.

		x_1	x_2	x_3	C_1	C_2	C_3	C_4
v_1	=	1	0	0	1	0	0	1
v_1'	=	1	0	0	0	1	1	0
v_2	=	0	1	0	0	0	0	1
v_2'	=	0	1	0	1	1	1	0
v_3	=	0	0	1	0	0	1	1
v_3'	=	0	0	1	1	1	0	0
s_1	=	0	0	0	1	0	0	0
s_1'	=	0	0	0	2	0	0	0
s_2	=	0	0	0	0	1	0	0
s_2'	=	0	0	0	0	2	0	0
s_3	=	0	0	0	0	0	1	0
s_3'	=	0	0	0	0	0	2	0
s_4	=	0	0	0	0	0	0	1
s_4'	=	0	0	0	0	0	0	2
t	=	1	1	1	4	4	4	4

Summary: subset-sum problem

- The 3-CNF formula ϕ is satisfiable if and only if the translated subset-sum problem is solvable.
- The essential insight is that subset-sum problem can encode a number of independent constraints (one constraint for each column). We then translate each variable and clause into one constraint.
- **Conclusion:** subset-sum problem is NP-complete.