Algorithms 2018: NP-Completeness

(Based on [Manber 1989])

Yih-Kuen Tsay

June 12, 2018

1 P vs. NP

P vs. NP

- P denotes the class of all problems that can be solved by *deterministic* algorithms in *polynomial* time.
- NP denotes the class of all problems that can be solved by *nondeterministic* algorithms in *polynomial* time.
- A nondeterministic algorithm, when faced with a choice of several options, has the power to guess the right one (if there is any).
- We will focus on *decision* problems, whose answer is either yes or no.

2 Polynomial-Time Reductions

Decision as Language Recognition

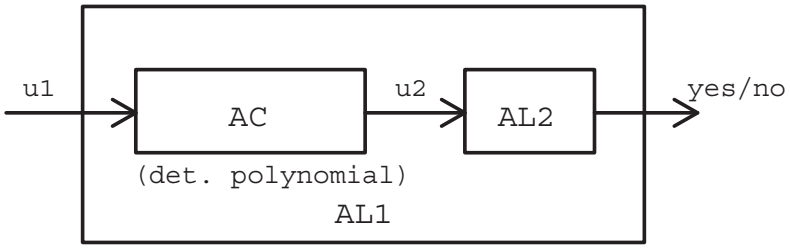
- A *decision* problem can be viewed as a *language-recognition* problem.
- Let U be the set of all possible inputs to the decision problem and $L \subseteq U$ be the set of all inputs for which the answer to the problem is yes.

  /* Every input is in essence a string. */
- We call L the *language* corresponding to the problem.

  /* A set of strings is conventionally called a language in mathematics and theory of computing, very much like a natural language which is essentially a set of meaningful sentences/strings in that language. */
- The decision problem is to recognize whether a given input belongs to L.

Polynomial-Time Reductions

- Let L_1 and L_2 be two languages from the input spaces U_1 and U_2.
- We say that L_1 is *polynomially reducible* to L_2 if there exists a conversion algorithm AC satisfying the following conditions:
  1. AC runs in polynomial time (deterministically).
  2. $u_1 \in L_1$ *if and only if* $AC(u_1) = u_2 \in L_2$.



Polynomial-Time Reductions (cont.)

Theorem 1 (11.1)**.** *If L_1 is polynomially reducible to L_2 and there is a polynomial-time algorithm for L_2, then there is a polynomial-time algorithm for L_1.*

/* Suppose f is an $O(n^{k_1})$-time conversion algorithm from L_1 to L_2, which has an $O(n^{k_2})$-time algorithm. Given an input w to L_1, the output $f(w)$ produced by f is at most $O(|w|^{k_1})$ long. An algorithm for L_1 starts by computing $f(w)$ (given w as input) and then runs the algorithm for L_2 on $f(w)$. These take at most $O(|w|^{k_1}) + O((|w|^{k_1})^{k_2}) = O(|w|^{k_1 \times k_2})$ time. */

Theorem 2 (11.2: transitivity)**.** *If L_1 is polynomially reducible to L_2 and L_2 is polynomially reducible to L_3, then L_1 is polynomially reducible to L_3.*

3 NP-Completeness

NP-Completeness

- A problem X is called an **NP-hard** problem if every problem in NP is polynomially reducible to X.
- A problem X is called an **NP-complete** problem if (1) X belongs to NP, and (2) X is NP-hard.

  Lemma 3 (11.3)**.** *A problem X is an NP-complete problem if (1) X belongs to NP, and (2′) Y is polynomially reducible to X, for some NP-complete problem Y.*

  /* Proving the NP-hardness of X from scratch, i.e., "every problem in NP is polynomially reducible to X", is in general very hard. The lemma makes it possible to consider just the reduction from an existing NP-hard problem to X. */
- If there exists an efficient (polynomial-time) algorithm for any NP-complete problem, then there exist efficient algorithms for all NP-complete (and hence all NP) problems.

4 The SAT Problem

The Satisfiability Problem (SAT)

Problem 4. *Given a Boolean expression in conjunctive normal form, determine whether it is satisfiable.*

- A Boolean expression is in *conjunctive normal form* (CNF) if it is the product of several sums, e.g., $(x + y + \bar{z}) \cdot (\bar{x} + y + z) \cdot (\bar{x} + \bar{y} + \bar{z})$.
- A Boolean expression is said to be *satisfiable* if there exists an assignment of 0s and 1s to its variables such that the value of the expression is 1.

SAT (cont.)

Theorem 5 (Cook's Theorem)**.** *The SAT problem is NP-complete.*

- This is our starting point for showing the NP-completeness of some other problems.
- Their NP-hardness will be proved by reduction directly or indirectly from SAT.

NP-Complete Problems

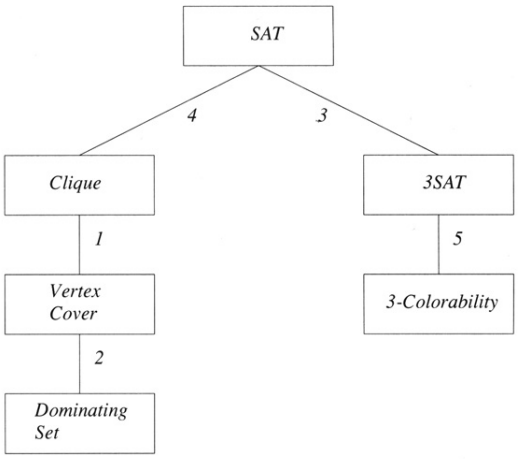


Figure 11.1 The order of NP-completeness proofs in the text.

Source: [Manber 1989].

5 Vertex Cover

Vertex Cover

Problem 6. *Given an undirected graph $G = (V, E)$ and an integer k, determine whether G has a vertex cover containing $\leq k$ vertices.*

A *vertex cover* of G is a set of vertices such that every edge in G is incident to at least one of these vertices.

Theorem 7 (11.4)**.** *The vertex-cover problem is NP-complete.*

Main idea: by reduction from the clique problem.

Vertex Cover (cont.)

Proof outline:

- The vertex-cover problem is in NP, since given a graph we can guess a subset of vertices and check whether it contains $\leq k$ vertices and is indeed a vertex cover in ploynomial time.
- The clique problem, which is NP-complete, is polynomially reducible to the vertex-cover problem.
  - Let $G(V, E)$ and k represent an arbitrary instance of the clique problem.
  - Let $\overline{G}(V, \overline{E})$ be the complement of G; computing the complement of a graph takes only polynomial time.
  - Claim: G has a clique of size $\geq k$ iff $\overline{G}$ has a vertex cover of size $\leq |V| - k$.

6 Dominating Set

Dominating Set

Problem 8. *Given an undirected graph $G = (V, E)$ and an integer k, determine whether G has a dominating set containing $\leq k$ vertices.*

A *dominating set* D is a set of vertices such that every vertex of G is either in D or is adjacent to some vertex in D.

Theorem 9 (11.5)**.** *The dominating-set problem is NP-complete.*

By reduction from the vertex-cover problem.
/* Below is a detailed proof, in which we make clearer certain conditions that are omitted or implicitly assumed in Manber's book. With the proof as an example, we also wish to clarify how the definition of polynomial-time reduction is followed in a typical NP-completeness proof.

The problem is obviously in NP, as we can guess a set of vertices and check in polynomial time whether the set is of size $\leq k$ and is indeed a dominating set of the given graph G. To prove that it is NP-hard, we demonstrate a polynomial-time reduction from the vertex-cover problem, which is known to be NP-hard. An input $(G_1 = (V_1, E_1), k_1)$, which is a pair of a graph and an integer, to the vertex cover problem can be converted to an input $(G_2 = (V_2, E_2), k_2)$ to the dominating set problem in the following manner:

To obtain G_2, we first remove all isolated vertices (which are not connected to any other vertex) from V_1. We then add, for each edge $\{u, v\}$ in E, a vertex uv and two edges $\{u, uv\}$ and $\{uv, v\}$. In other words, we transform every edge into a triangle. Finally, we make k_2 simply equal to k_1. This conversion apparently can be done by a deterministic algorithm in polynomial time.

We need to show that G_1 has a vertex cover of size $\leq k_1$ if and only if G_2 has a dominating set of size $\leq k_2$. But before doing so, we deviate to make a contrast with the definition of polynomial-time reduction (which can be found in the appendix). The input spaces U_{vc} and U_{ds} of the two problems are the same, namely the set of all possible pairs of a graph and an integer. The language L_{vc} of the vertex cover problem is the set of all (G, k) such that G has a vertex cover of size $\leq k$, while the language L_{ds} of the dominating set problem is the set of all (G, k) such that G has a dominating set of size $\leq k$. The proof obligation "G_1 has a vertex cover of size $\leq k_1$ if and only if G_2 has a dominating set of size $\leq k_2$" is derived from the statement "$(G_1, k_1) \in L_{vc}$ iff $(G_2, k_2) \in L_{ds}$". (End of Deviation)

The "only if" part: Suppose G_1 has a vertex cover C of size $\leq k_1$. Remove all isolated vertices in C to obtain another vertex cover C' of G_1 (isolated vertices are not usual for covering an edge). C' is also a subset of V_2 and $|C'| \leq |C| \leq k_1 = k_2$. We claim that C' is a dominating set of G_2. Every vertex u in V_2 that comes from V_1 is an end vertex of some edge $\{u, v\} \in E_1$. Since $\{u, v\}$ is covered by C', either u or v must be in C', implying that u is dominated by C', i.e., u is either in C' or adjacent to some vertex (namely v) in C'. Every new vertex uv that was added for edge $\{u, v\}$ is adjacent to both u and v and is also dominated, as again one of u and v must be in C'.

The "if" part: Suppose G_2 has a dominating set D of size $\leq k_2$. D may not be a subset of V_1, as D may contain vertices that were added in the conversion. Replace every vertex uv in D, which was added for edge $\{u, v\}$, by either u or v to obtain a new set D'. Since every replaced vertex is adjacent to the replacing vertex, D' remains a dominating set of G_2. D' is a subset of V_1 and $|D'| \leq |D| \leq k_2 = k_1$ ($|D'|$ is not necessarily equal to $|D|$). We claim that D' is also a vertex cover of G_1. For every edge $\{u, v\}$ in E_1, either u or v is in D'; otherwise, the added vertex uv in V_2 corresponding to $\{u, v\}$ would not be dominated by D'. Therefore, every edge of G_1 is covered by D'. */

Dominating Set (cont.)

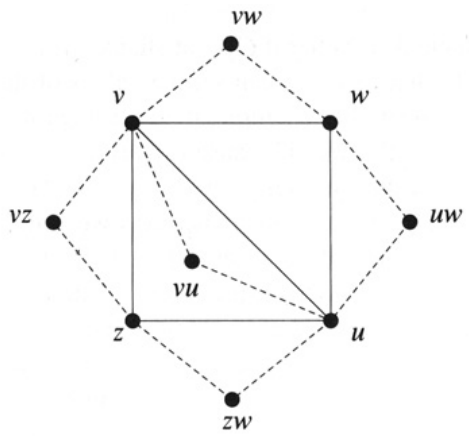


Figure 11.2 The dominating-set reduction.

(Note: zw should be zu in the graph.)
Source: [Manber 1989].

7 3SAT

3SAT

Problem 10. *Given a Boolean expression in CNF such that each clause contains exactly three variables, determine whether it is satisfiable.*

Theorem 11 (11.6)**.** *The 3SAT problem is NP-complete.*

By reduction from the regular SAT problem.

3SAT (cont.)

- From an arbitrary clause $(x_1 + x_2 + \cdots + x_k)$, where $k \neq 3$, of the SAT problem to clauses of the 3SAT problem:
  - When $k \geq 4$,
$$\begin{aligned}(x_1 + x_2 + y_1)\cdot\\ (x_3 + \overline{y_1} + y_2)\cdot\\ (x_4 + \overline{y_2} + y_3)\cdot\\ \vdots\\ (x_{k-2} + \overline{y_{k-4}} + y_{k-3})\cdot\\ (x_{k-1} + x_k + \overline{y_{k-3}})\end{aligned}$$
  - When $k = 2$,
$$(x_1 + x_2 + w) \cdot (x_1 + x_2 + \overline{w})$$
  - When $k = 1$,
$$(x_1 + y + z) \cdot (x_1 + \overline{y} + z) \cdot (x_1 + y + \overline{z}) \cdot (x_1 + \overline{y} + \overline{z})$$

  /* The new variables must be distinct across the different original clauses. */

8 Clique

Clique

Problem 12. *Given an undirected graph $G = (V, E)$ and an integer k, determine whether G contains a clique of size $\geq k$.*

A *clique* C is a subgraph of G such that all vertices in C are adjacent to all other vertices in C.

Theorem 13 (11.7)**.** *The clique problem is NP-complete.*

By reduction from the SAT problem.

Clique (cont.)

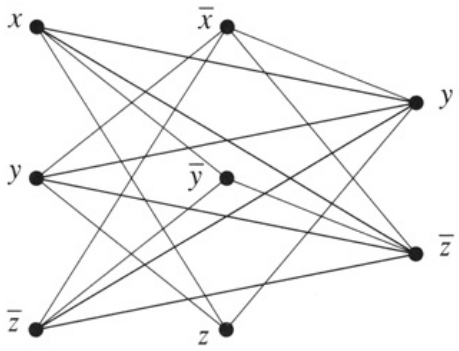


Figure 11.3 An example of the clique reduction for the expression $(x + y + \overline{z}) \cdot (\overline{x} + \overline{y} + z) \cdot (y + \overline{z})$.

Source: [Manber 1989].

9 3-Coloring

3-Coloring

Problem 14. *Given an undirected graph $G = (V, E)$, determine whether G can be colored with three colors.*

Theorem 15 (11.8)**.** *The 3-coloring problem is NP-complete.*

By reduction from the 3SAT problem.

3-Coloring (cont.)

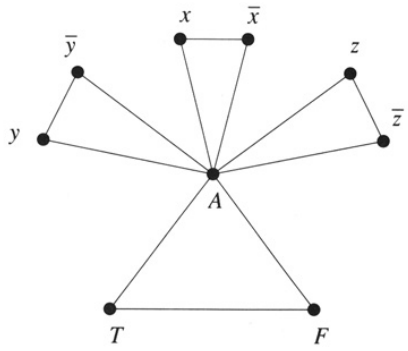


Figure 11.4 The first part of the construction in the reduction of 3SAT to 3-coloring.

Source: [Manber 1989].

3-Coloring (cont.)

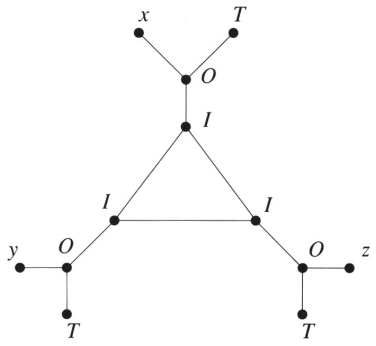


Figure 11.5 The subgraphs corresponding to the clauses in the reduction of 3SAT to 3-coloring.

Source: [Manber 1989].

3-Coloring (cont.)

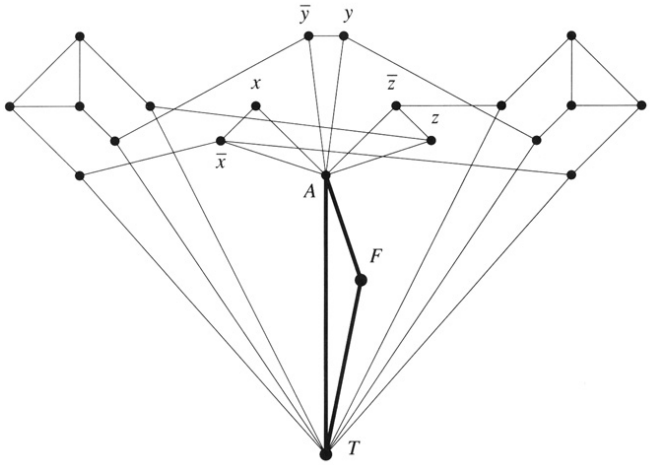


Figure 11.6 The graph corresponding to $(\bar{x}+y+\bar{z})\cdot(\bar{x}+\bar{y}+z)$.

Source: [Manber 1989].

10 More NP-Complete Problems

More NP-Complete Problems

- **Independent set**: An independent set in an undirected graph is a set of vertices no two of which are adjacent. The problem is to determine, given a graph G and an integer k, whether G contains an independent set with $\geq k$ vertices.
- **Hamiltonian cycle**: A Hamiltonian cycle in a graph is a (simple) cycle that contains each vertex exactly once. The problem is to determine whether a given graph contains a Hamiltonian cycle.

More NP-Complete Problems (cont.)

- **Travelling salesman**: The input includes a set of cities, the distances between all pairs of cities, and a number D. The problem is to determine whether there exists a (travelling-salesman) tour of all the cities having total length $\leq D$.
- **Partition**: The input is a set X where each element $x \in X$ has an associated size $s(x)$. The problem is to determine whether it is possible to partition the set into two subsets with exactly the same total size.

More NP-Complete Problems (cont.)

- **Knapsack**: The input is a set X, where each element $x \in X$ has an associated size $s(x)$ and value $v(x)$, and two other numbers S and V. The problem is to determine whether there is a subset $B \subseteq X$ whose total size is $\leq S$ and whose total value is $\geq V$.
- **Bin packing**: The input is a set of numbers $\{a_1, a_2, \cdots, a_n\}$ and two other numbers b and k. The problem is to determine whether the set can be partition into k subsets such that the sum of numbers in each subset is $\leq b$.