Short Introduction to $\mathbf{NP}$-completeness

Hung Q. Ngô *

November 27, 2007

Abstract

This document is still under construction and is far from being complete. However, it consists of many of the things we have discussed in class. Read it with caution.

1 Proving $\mathbf{NP}$-completeness results

Given SAT as a basis, Lemma **??** helps show many other problems are $\mathbf{NP}$-complete. To show that a problem Π is $\mathbf{NP}$-complete, we first have to show that $\Pi \in \mathbf{NP}$, which in most cases is trivial; then, we find another problem Π' which has been shown to be $\mathbf{NP}$-complete, and show that $\Pi' \leq_p \Pi$. Note the direction of reduction here. Many students have done the reduction in the wrong direction, due to the somewhat confusing nature of the word "reducible" in this context.

Another point worth noticing is that finding a reduction often helps us understand the structure of the problem at hand much better. In effect, we show that our problem is a special case of another known problem. If there was a good approximation algorithm to solve the other problem, then our problem can benefit from it via the reduction.

Beside the problems defined earlier like VC, CLIQUE, IS, ..., we define a few more problems here and show that they are also $\mathbf{NP}$-complete. The proofs shall also illustrate several $\mathbf{NP}$-completeness proof techniques. More examples and a classification of techniques can be found in the classic text by Garey and Johnson [3].

3-SAT

INSTANCE: A collection $\mathcal{C}$ of clauses $\{C_1, \ldots, C_m\}$ over $X = \{x_1, \ldots, x_n\}$, where each clause C_i consists of exactly 3 literals.

QUESTION: Is there a truth assignment satisfying all clauses in $\mathcal{C}$.

SET COVER (SC)

INSTANCE: A family $\mathcal{S}$ of subsets $\{S_1, \ldots, S_m\}$ of a finite universe U ($|U| = n$), and a bound $b \in \mathbb{Z}^+$.

QUESTION: Is there $I \subseteq \{1, \ldots, m\}$, $|I| \leq b$, such that

$$U = \bigcup_{i \in I} S_i.$$

The collection $\{S_i \mid i \in I\}$ satisfying the above condition is called a *set cover* of U.

DOMINATING SET (DS)

*Computer Science and Engineering Department, SUNY at Buffalo, Amherst, NY 14260, USA. E-mail: hungngo@cse.buffalo.edu.

INSTANCE: A graph $G = (V, E)$, a bound $b \in \mathbb{Z}^+$.

QUESTION: Is there a subset $S \subseteq V$, $|S| \leq b$, such that every vertex v not in S is incident to some vertex in S. The vertices in S *dominates* all vertices in V.

3-COLORABILITY

INSTANCE: A graph $G = (V, E)$.

QUESTION: Is G 3-*colorable*, i.e. is there a way to assign each vertex of G one of 3 colors such that two adjacent vertices have different colors.

k-COLORABILITY

INSTANCE: A graph $G = (V, E)$, $k \geq 3$.

QUESTION: Is G k-*colorable*, i.e. is there a way to assign each vertex of G one of k colors $\{1, \dots, k\}$ such that two adjacent vertices have different colors.

HAMILTONIAN CYCLE (HC)

INSTANCE: A graph $G = (V, E)$.

QUESTION: Does G contain a Hamiltonian cycle? (An HC is a cycle containing all vertices of G.)

TRAVELING SALESMAN PROBLEM (TSP)

INSTANCE: A complete graph $G = (V, E)$, a cost function $c : E \to \mathbb{Z}^+$, and a bound $b \in \mathbb{Z}^+$.

QUESTION: Is there a TSP tour with total cost at most b?

EUCLIDEAN TRAVELING SALESMAN (EUCLIDEAN TSP)

INSTANCE: A set $C = \{p_1, \dots, p_n\}$ of n points on a plane given by their x, y-coordinates $p_i = (x_i, y_i) \in \mathbb{Z} \times \mathbb{Z}$, a bound $b \in \mathbb{Z}^+$.

QUESTION: Is there a TSP tour with total length at most b?

SUBSET SUM (SS)

INSTANCE: A finite set S of natural numbers, and a target $t \in \mathbb{N}$.

QUESTION: Is there a subset $S' \subseteq S$, whose elements sum up to t?

KNAPSACK

INSTANCE: n items, their values $v_i \in \mathbb{Z}^+$, their corresponding weights $w_i \in \mathbb{Z}^+$, a weight limit $W \in \mathbb{Z}^+$, and a value limit $V \in \mathbb{Z}^+$.

QUESTION: Is there a subset of items with total weight at most W, and total value at least V?

Figure 1 is our road map of the reductions to show all aforementioned problems are $\mathbf{NP}$-complete. The arrows point to the direction of reduction. We shall present many of the reductions shown, and left a few easier ones as exercises in the next section.

Theorem 1.1. *3-SAT is* $\mathbf{NP}$*-complete.*

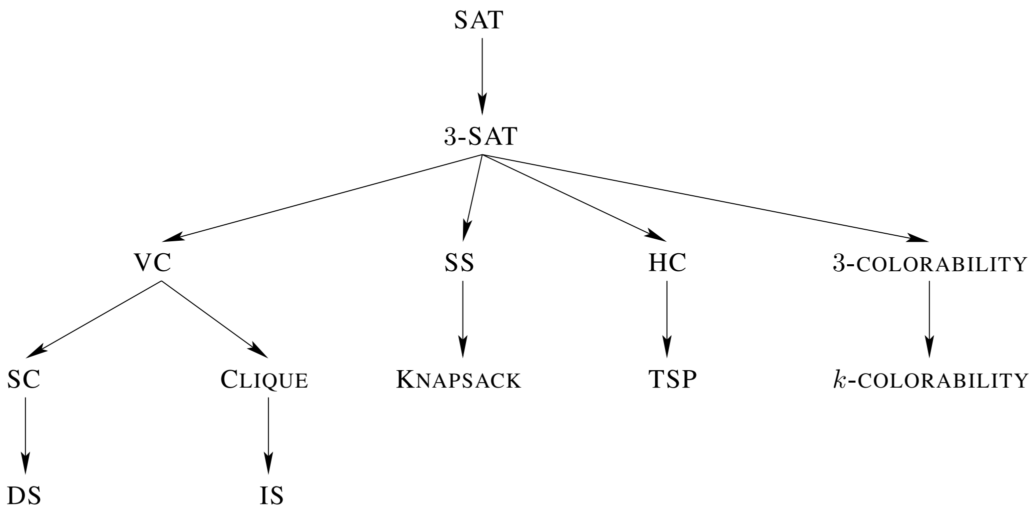


Figure 1: Map of our reductions to show $\mathbf{NP}$-complete problems.

Proof. 3-SAT is in $\mathbf{NP}$ because given a truth assignment $t : X \to \{T, F\}$, we can easily check if the given boolean function f gets value TRUE under t. All one has to do is to check, for each clause C_i, if one of the 3 literals in C_i was assigned T.

To complete the proof, we show that SAT $\leq_p$ 3-SAT. Given an instance $\mathcal{C} = \{C_1, \ldots, C_m\}$ of SAT, we would like to construct an instance $\mathcal{C}' = \{C'_1, \ldots, C'_{m'}\}$ of 3-SAT in polynomial time such that $\mathcal{C}$ is satisfiable if and only if $\mathcal{C}'$ is satisfiable.

Suppose $C_i = \{y_{i_1}, \ldots, y_{i_{j_i}}\}$, namely there are j_i literals in C_i indexed from i_1 to i_{j_i}. For each clause C_i of $\mathcal{C}$ we construct a few clauses of $\mathcal{C}'$, each with exactly 3 literals. The union of the clauses constructed from the C_i would form $\mathcal{C}'$.

We distinguish several cases.

The first case is when C_i has only one literal y_{i_1}. The corresponding clauses of $\mathcal{C}'$ are

$$\{\mathbf{y_{i_1}}, z_{i_1}, z_{i_2}\}, \{\mathbf{y_{i_1}}, z_{i_1}, \bar{z}_{i_2}\}, \{\mathbf{y_{i_1}}, \bar{z}_{i_1}, z_{i_2}\}, \{\mathbf{y_{i_1}}, \bar{z}_{i_1}, \bar{z}_{i_2}\}.$$

The variables z_{i_k} are auxiliary variables, which is not in the set X of variables of $\mathcal{C}$.

The second case is when $C_i = \{y_{i_1}, y_{i_2}\}$. The corresponding clauses are

$$\{\mathbf{y_{i_1}}, \mathbf{y_{i_2}}, z_i\}, \{\mathbf{y_{i_1}}, \mathbf{y_{i_2}}, \bar{z}_i\}.$$

The third case is when C_i has exactly 3 literals, in which case we include C_i in $\mathcal{C}'$.

The last case is when $C_i = \{y_{i_1}, y_{i_2}, \ldots, y_{i_k}\}$, $k \geq 4$. The corresponding clauses are

$$\{\mathbf{y_{i_1}}, \mathbf{y_{i_2}}, z_{i_1}\}, \{\bar{z}_{i_1}, \mathbf{y_{i_3}}, z_{i_2}\}, \{\bar{z}_{i_2}, \mathbf{y_{i4}}, z_{i_3}\}, \ldots, \{\bar{z}_{i_{k-3}}, \mathbf{y_{i_{k-1}}}, \mathbf{y_{i_k}}\}.$$

For example, if

$$\mathcal{C} = \{\{\bar{x}_3\}, \{x_1, \bar{x}_4\}, \{x_1, x_3, x_4\}, \{x_1, \bar{x}_2, \bar{x}_3, x_4, x_5, \bar{x}_6\}\},$$

then

$$\begin{aligned}
\mathcal{C}' &= \big\{\{\bar{\mathbf{x}}_\mathbf{3}, a, b\}, \{\bar{\mathbf{x}}_\mathbf{3}, \bar{a}, b\}, \{\bar{\mathbf{x}}_\mathbf{3}, a, \bar{b}\}, \{\bar{\mathbf{x}}_\mathbf{3}, \bar{a}, \bar{b}\}\big\} \\
&\bigcup \ \{\{\mathbf{x_1}, \bar{\mathbf{x}}_\mathbf{4}, c\}\{\mathbf{x_1}, \bar{\mathbf{x}}_\mathbf{4}, \bar{c}\}\} \\
&\bigcup \ \{\{\mathbf{x_1}, \mathbf{x_3}, \mathbf{x_4}\}\} \\
&\bigcup \ \big\{\{\mathbf{x_1}, \bar{\mathbf{x}}_\mathbf{2}, d_1\}\{\bar{d}_1, \bar{\mathbf{x}}_\mathbf{3}, d_2\}\{\bar{d}_2, \mathbf{x_4}, d_3\}\{\bar{d}_3, \mathbf{x_5}, \bar{\mathbf{x}}_\mathbf{6}\}\big\}.
\end{aligned}$$

To this end, we show that this construction yields a polynomial time reduction. The fact that this construction could be done in poly-time is obvious. We are left to show that $\mathcal{C}$ is satisfiable if and only if $\mathcal{C}'$ is satisfiable. To show this, it is sufficient to show that a particular clause C_i is satisfiable if and only if its corresponding size-3 clauses in $\mathcal{C}'$ are satisfiable.

For the forward direction, let assume that C_i is satisfiable with some truth assignment t. If $C_i = \{y_{i_1}\}$, then we could assign any value to the z_{i_j}, while making the four corresponding clauses in $\mathcal{C}'$ TRUE. When $C_i = \{y_{i_1}, y_{i_2}\}$, assume without loss of generality that the literal y_{i_1} is TRUE under t, then we assign $\bar{z}_i$ to be TRUE and the two corresponding clauses $\{\mathbf{y_{i_1}}, \mathbf{y_{i_2}}, z_i\}$ and $\{\mathbf{y_{i_1}}, \mathbf{y_{i_2}}, \bar{z}_i\}$ are also TRUE. If C_i contains exactly three literals, then we can keep the truth assignment the same. Lastly, when C_i has four or more literals, without loss of generality we assume some $y_{i_{j_0}}$ is TRUE. In this case, we can "propagate" to both sides the TRUE/FALSE assignments of the z_{i_j}, namely we assign $z_{i_{j_0-2}}, \ldots, z_{i_1}$ to be TRUE, and $\bar{z}_{i_{j_0-1}}, \ldots, \bar{z}_{i_{k-3}}$ to be TRUE.

Conversely, suppose $\mathcal{C}'$ is satisfiable with a truth assignment t'. A simple case by case analysis implies that the same assignment restricted only to the y's satisfies $\mathcal{C}$. Let us consider one of the cases, the reader is encouraged to complete the rest of the proof. Consider the four clauses corresponding to some C_i of size one:

$$\{\mathbf{y_{i_1}}, z_{i_1}, z_{i_2}\}, \{\mathbf{y_{i_1}}, z_{i_1}, \bar{z}_{i_2}\}, \{\mathbf{y_{i_1}}, \bar{z}_{i_1}, z_{i_2}\}, \{\mathbf{y_{i_1}}, \bar{z}_{i_1}, \bar{z}_{i_2}\}.$$

If these four clauses are satisfiable, then some pair of z_{i_1}, z_{i_2} and their negations have to be both FALSE, in which case the corresponding clause is TRUE if and only if y_{i_1} was assigned TRUE. □

Theorem 1.2. VERTEX COVER *is* $\mathbf{NP}$*-complete.*

Proof. VERTEX COVER is in $\mathbf{NP}$, since given a subset of vertices we can easily verify that it is a vertex cover of size at most b.

To show that VC is $\mathbf{NP}$-complete, we will prove 3-SAT $\leq_p$ VC. Given an instance $\mathcal{C} = \{C_1, \ldots, C_m\}$ of 3-SAT on n variables $X = \{x_1, \ldots, x_n\}$, we will construct a graph G in polynomial time such that $\mathcal{C}$ is satisfiable if and only if G has a vertex cover of size at most $2m + n$. Figure illustrates our construction. The graph G consists of exactly $2n + 3m$ vertices, $2n$ of which correspond to $2n$ literals

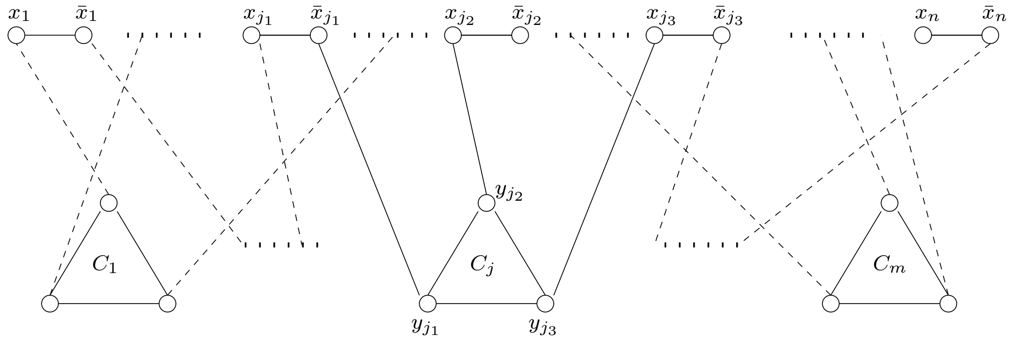


Figure 2: 3-SAT $\leq_p$ VERTEX COVER. Assume $C_j = \{\bar{x}_{j_1}, x_{j_2}, x_{j_3}\}$.

$x_1, \bar{x}_1, \ldots, x_n, \bar{x}_n$, and the other $3m$ vertices correspond to m clauses. Each clause is represented by a triangle in the graph. There is an edge between each vertex labelled x_i and its negation $\bar{x}_i$. If clause C_j consists of literals $y_{j_1}, y_{j_2}, y_{j_3}$, then the tree vertices of the triangle corresponding to C_j will be connected to the three vertices corresponding to the three literals. It is easy to see that the construction of G can be done in polynomial time.

We now show that $\mathcal{C}$ is satisfiable iff G has a vertex cover of size at most $2m + n$.

Suppose first that $\mathcal{C}$ is satisfiable with some truth assignment $t : X \to \{\text{TRUE}, \text{FALSE}\}$. We can construct a vertex cover S of G of size $n + 2m$ as follows. If $t(x_i) = $ TRUE, then vertex x_i is in S, otherwise vertex $\bar{x}_i$ is in S. This accounts for n vertices of S. For each clause C_j, one of its literals, say y_{j_1}, is TRUE since C_j is satisfied by t. We put the other two vertices of the triangle corresponding to y_{j_2} and y_{j_3} in S. This accounts for the other $2m$ vertices of S. To see that S is a vertex cover, first consider the edges connecting a variable x_i and its negation $\bar{x}_i$. Either x_i or $\bar{x}_i$ is in S, hence this edge is covered. Second, consider the edges in the triangles. Two out of three vertices in each triangle are picked, hence all three edges in the triangle are covered by S. Lastly, consider the three edges connecting a triangle C_j and the $x_i, \bar{x}_i$ vertices. Two out of these three edges are covered by the two vertices in S of the C_j-triangle. The other edge is covered since the other literal is in S.

Conversely, suppose there is a vertex cover S of G of size at most $n + 2m$. In order to cover three edges of a triangle, at least two vertices in the triangle have to be in S. In order to cover the $(x_i, \bar{x}_i)$ edge, at least one of x_i and $\bar{x}_i$ has to be in S. Hence, in fact $|S| = n+2m$ and S consists of exactly two vertices out of each triangle and one out of each pair $x_i, \bar{x}_i$. Consider a truth assignment $t : X \to \{\text{TRUE}, \text{FALSE}\}$ where $t(x_i) = \text{TRUE}$ if vertex x_i is in S, and $t(x_i) = \text{FALSE}$ if vertex $\bar{x}_i$ is in S. We claim that all clauses are satisfied by this truth assignment. Consider a clause C_j. Let's look at the three edges coming out of triangle C_j. The two vertices in S of this triangle can only cover two of these edges. Hence, the other edge has to be covered by a vertex corresponding to the other literal, say y_{j_1}, which means $y_{j_1} = \text{TRUE}$ under t, i.e. C_j is satisfied. □

Remark 1.3. The **NP**-completeness proofs of 3-SAT and VC were presented in their full rigor. You can see that the actual ideas were much simpler than the lengths of the proofs would otherwise indicate. From this point on, I will present proofs so that they convey the main ideas without too many notational details. You should follow the formats of the previous two proofs when working on homework problems. The wording can be shorter, as long as the main idea gets across.

Corollary 1.4. *Both* CLIQUE *and* IS *are* **NP***-complete.*

Proof. This follows easily from the reductions we have done in Section **??** and the previous theorem. (One should not forget to check that CLIQUE and IS are in **NP**. □

Theorem 1.5. SUBSET SUM *is* **NP***-complete.*

Proof. SUBSET SUM is obviously in **NP**. We will show that 3-SAT $\leq_p$ SUBSET SUM. Consider an instance $\mathcal{C} = \{C_1, \dots, C_m\}$ of 3-SAT on n variables $X = \{x_1, \dots, x_n\}$. We will construct an instance of SUBSET SUM as follows.

The set S of natural numbers consists of $2n + 2m$ numbers:

- For each literal y there corresponds a member $a^{(y)}$ of S whose binary representation consists of $n+3m$ bits. Suppose y is either x_i or $\bar{x}_i$ for some $i \in \{1, \dots, n\}$. Write $a^{(y)} = a^{(y)}_{n+3m} a^{(y)}_{n+3m} \dots a^{(y)}_1$. The n most significant bits are all 0, except for $a^{(y)}_{n+3m-i+1} = 1$. If literal y appears in clause C_j, then $a^{(y)}_{3j-2} = 1$. The rest of the bits of $a^{(y)}$ are 0.
- There are $2m$ other "auxiliary" numbers $a^{(j,1)}$ and $a^{(j,2)}$ for $j = 1, \dots, m$. In binary format, these numbers have $3m$ bits each. Write $a^{(j,1)} = a^{(j,1)}_{3m} \dots a^{(j,1)}_1$ and $a^{(j,2)} = a^{(j,2)}_{3m} \dots a^{(j,2)}_1$, then only $a^{(j,1)}_{3j-2} = 1$ and $a^{(j,2)}_{3j-1} = 1$, while the rest of their bits are 0.

The target t is also a $(n + 3m)$-bit binary number. The n most significant bits of t are all 1. The last $3m$ bits go $100100 \dots 100$.

We claim that $\mathcal{C}$ has a satisfying truth assignment if and only if there is a subset of numbers in S whose sum is t. The basic idea is as follows.

We have to make sure that we cannot take $a^{(x_i)}$ and $a^{\bar{x}_i}$ at the same time. This is guaranteed by the fact that the n most significant bits of t are all 1. If we take both $a^{(x_i)}$ and $a^{\bar{x}_i}$, then the corresponding bit in the sum will be 0 (at least for the largest i for which both $a^{(x_i)}$ and $a^{\bar{x}_i}$ are taken).

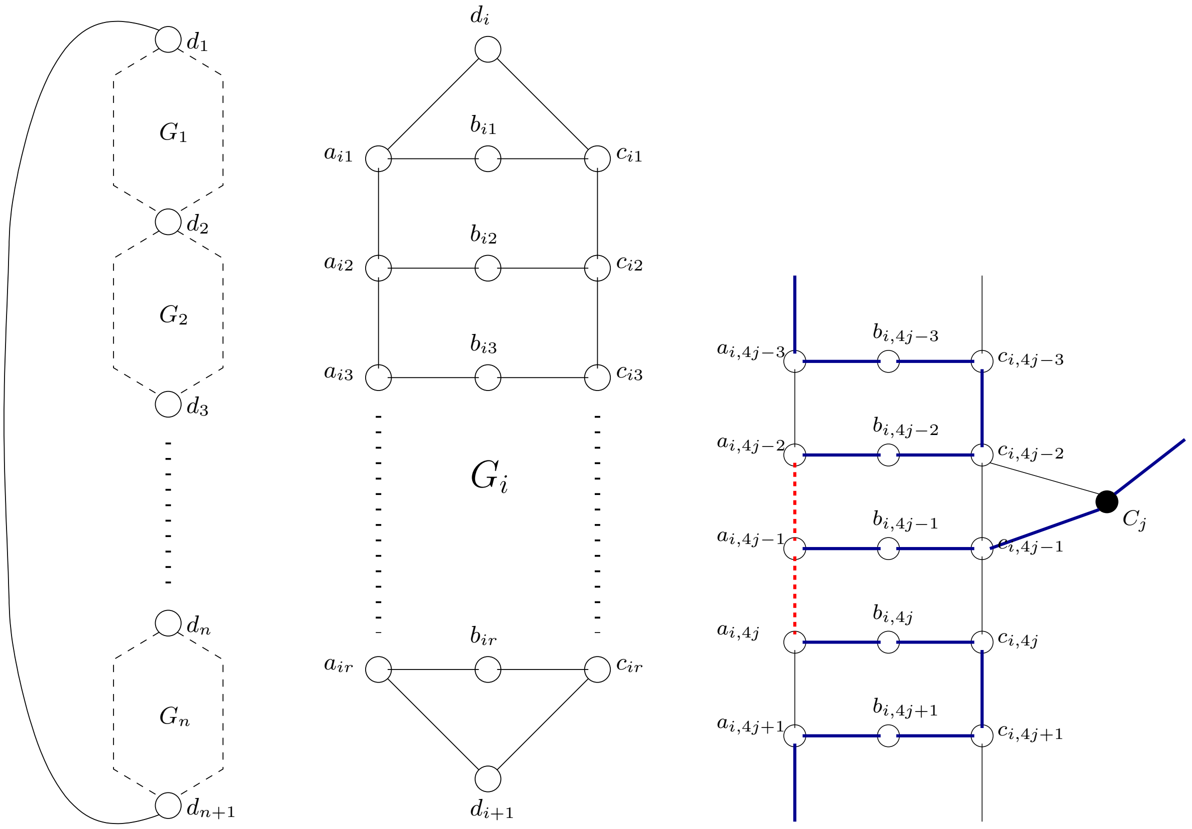
The $3m$ least significant bits are divided into blocks of size 3 each, where the jth block (from the right) is used to verify that C_j is satisfied. If some $a^{(y)}$ is chosen, then there is a 1 in all the blocks corresponding the C_j in which y occurs, and C_j is satisfied. However, C_j can be satisfied by one, two, or three of its literal. The corresponding block in t is 100, representing number 4. If C_j is satisfied by 1 literal, then we can pick up both $a^{(j,1)}$ and $a^{(j,2)}$ so that the sum in the block is 4. If C_j is satisfied by 2 or 3 literals, then we pick either $a^{(j,2)}$ or $a^{(j,1)}$ to "fill in." If C_j is not satisfied, then the sum of $a^{(j,2)}$ and $a^{(j,1)}$ is not sufficient to make the result 4.

Lastly, notice that even when C_j is satisfied by 3 literals, and when both $a^{(j,2)}$ and $a^{(j,1)}$ are picked, the total is only 7 – or 111 in binary. Hence, there cannot be any carry-over bit from the $3m$ least significant bits to the higher order bits.

The rest of the proof follows straightforwardly from the above analysis. □

Theorem 1.6. HAMILTONIAN CYCLE *is* **NP**-*complete.*

Proof. HC is obviously in **NP**. We will show that 3-SAT $\leq_p$ HC with a reduction by Du and Ko [2]. Given an instance $\mathcal{C} = \{C_1, \dots, C_m\}$ of 3-SAT on n variables $x_1, \dots x_n$, we construct a graph G which has a Hamiltonian cycle iff $\mathcal{C}$ is satisfiable. Figure illustrates our construction.



(a) General view (b) Component G_i corresponds to x_i (c) Connect to C_j

Figure 3: 3-SAT $\leq_p$ HAMILTONIAN CYCLE.

The construction goes as follows. The graph G consists of a series of components $G_1, \dots, G_n$ which corresponds to variables $x_1, \dots, x_n$ as shown in Figure 3(a). Each of G_i looks that that in Figure 3(b), where $r = 4m$.

Consider the G_i component. There are exactly two possible paths from d_i to d_{i+1} which cover all vertices of the component. Either we enter with (d_i, a_{i1}) or with (d_i, b_{i1}). The rest of such path is determined by the first move: just go zig-zag'ing to the end. Hence, there are 2^n Hamiltonian paths through G, corresponding to 2^n possible truth assignments to the variables x_i. If a path enters G_i at a_{i1}, then consider x_i to be TRUE, otherwise $\bar{x}_i$ is TRUE.

Note that a Hamiltonian path from d_i to d_{i+1} corresponding to setting $x_i = \text{TRUE}$ must pass through edge $(a_{i,4j-2}, a_{i,4j-1})$, for all $j = 1, \dots, m$.

Now, we add m more vertices labelled $C_1, \dots, C_m$. If C_j contains x_i, then we add two edges from C_j to $a_{i,4j-2}$ and $a_{i,4j-1}$. If C_j contains $\bar{x}_i$, then we add two edges from C_j to $c_{i,4j-2}$ and $c_{i,4j-1}$. Thus, each C_j has degree exactly 6.

The crucial point to notice is that, we cannot construct a Hamiltonian cycle which passes through some C_j from a vertex in some G_i to a vertex in some $G_{i'}$ with $i \neq i'$. In other words, in order to cover C_j we must go through vertices as if the C_j didn't exist, and then replace "locally" some edge $(a_{i,4j-2}, a_{i,4j-1})$ or $(c_{i,4j-2}, c_{i,4j-1})$ by "stopping by" C_j in between. To see that this claim is true, suppose – as shown in Figure 3(c) – that we passes through C_j from some other component and enters G_i at $c_{i,4j-1}$, say. Note that any Hamiltonian cycle has to pass through the horizontal edges to cover the $b_{p,q}$. Hence, both edges $(c_{i,4j-1}, b_{i,4j-1})$ and $(b_{i,4j-1}, a_{i,4j-1})$ are in the current path. Moreover, it is easy to see that both $(c_{i,4j-2}, c_{i,4j-3})$ and $(c_{i,4j}, c_{i,4j+1})$ are in the cycle. We now have a dilemma at vertex $a_{i,4j-1}$ since we do not know where to go from it. Note that, x_i and $\bar{x}_i$ cannot both be in C_j.

The rest of the proof now follows straightforwardly. A Hamiltonian cycle corresponds to a truth assignment where occasionally the cycle "stop by" C_j to say that my assignment satisfies C_j. The reader should have no trouble completing the proof. □

Theorem 1.7. 3-COLORABILITY *is* **NP**-*complete.*

Proof. It's obvious that 3-COLORABILITY is in **NP**. We will reduce 3-SAT to it. Given an instance $\mathcal{C} = \{C_1, \dots, C_m\}$ of 3-SAT on n variables $X = \{x_1, \dots, x_n\}$, we construct an instance of 3-COLORABILITY as follows. Figure illustrates our construction. There is a *top vertex* v. For each variable

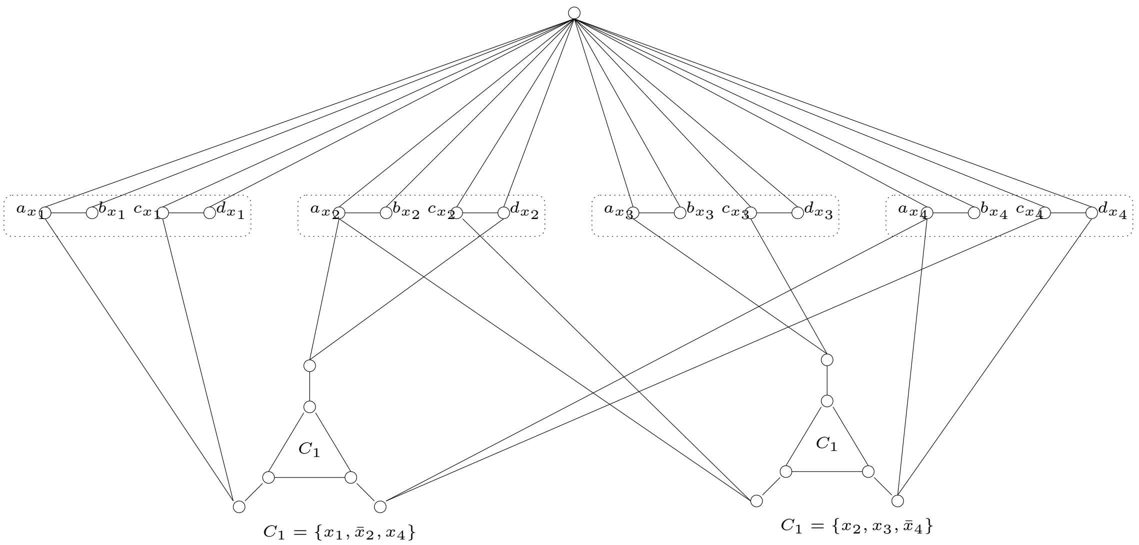


Figure 4: 3-SAT $\leq_p$ 3-COLORABILITY.

x_i, create 4 vertices $a_{x_i}, b_{x_i}, c_{x_i}$, and d_{x_i}. For each clause C_j, create a triangle with three extra edges pointing out to three extra vertices as shown in the figure. Each extra vertex corresponds to one of the three literals y_{j_1}, y_{j_2}, or y_{j_3} of the clause. If the literal is not a negation, say the literal is x_i, then connect the extra vertex x_i (of the clause' triangle) to a_{x_i} and c_{x_i}. If the literal is $\bar{x}_i$, then connect it to a_{x_i} and d_{x_i}.

We will show that $\mathcal{C}$ is satisfiable iff G is 3-colorable.

The main idea is that, for each triangle C_j, if the tree extra vertices get the same color then there is no way to color the inner three vertices with three colors. On the other hand, if there are at least two outside colors, then the inner vertices can always be colored.

Suppose first that $\mathcal{C}$ is satisfiable with truth assignment t. Color the top vertex v with color 1. If $t(x_i) = \text{TRUE}$ then color a_{x_i} and c_{x_i} with 2, and b_{x_i} and d_{x_i} with 3. If $t(x_i) = \text{FALSE}$ then color a_{x_i} and d_{x_i} with 2, and b_{x_i} and c_{x_i} with 3. The idea is that if a literal, say x_i, is true then there will be two choices of colors for the triangles' extra vertices which correspond to the literal, because a_{x_i} and c_{x_i} get the same color. Because all C_j are satisfiable, each C_j-triangle has at least one outside vertex with two color choices. Thus the outside vertices of C_j can be colors so that at least two colors are used. Then, the inner vertices of C_j can be properly colored due to the observation above.

Conversely, suppose the graph G is 3-colorable, we can find a truth assignment by assigning x_i to be TRUE if a_{x_i} and c_{x_i} get the same color, and FALSE otherwise. It is easy to check that this assignment satisfies all of the clauses. Consider a clause C_j, for instance. If C_j is not satisfiable then all three outside vertices of its triangle must have the same color as the top vertex, which means the inner vertices of C_j were not properly colored.

□

2 Exercises

Exercise 1. Show that SET COVER is **NP**-complete.

Exercise 2. Show that DOMINATING SET is **NP**-complete.

Exercise 3. Show that KNAPSACK is **NP**-complete.

Exercise 4. Show that TSP is **NP**-complete.

Exercise 5. Show that k-COLORABILITY is **NP**-complete.

For the following exercises, assume that you only know the problems in our roadmap are **NP**-complete.

Exercise 6. Give a polynomial time reduction from 3-SAT to CLIQUE.

Exercise 7 (**)**.** Give a polynomial time reduction from HAMILTONIAN CYCLE to VERTEX COVER.

Exercise 8 (*)**.** A *monotone boolean formula* is a boolean formula which involves only AND's and OR's. There is no negation. For instance, $\phi = (x_1 \wedge x_2) \vee (x_3 \vee (x_4 \wedge x_5))$ is monotone. Prove that the following problem is **NP**-complete.

MONOTONE SATISFIABILITY (MON-SAT)

> INSTANCE: A set X of variables, a monotone boolean formula ϕ over X, and a bound $b \in \mathbb{Z}^+$.
>
> QUESTION: Is there a truth assignment which satisfies ϕ in which at most b variables are TRUE?

Exercise 9. Prove that the following problem is **NP**-complete.

LONGEST PATH

> INSTANCE: A graph G and a bound $b \in \mathbb{Z}^+$.
>
> QUESTION: Is there a path in G of length at least b?

Exercise 10. Prove that the following problem is **NP**-complete.

HITTING SET

INSTANCE: A collection $\mathcal{C} = \{S_1, \dots, S_m\}$ of non-empty subsets of a universe set S, and a bound $b \in \mathbb{Z}^+$.

QUESTION: Is there a subset $H \subseteq S$ with $|H| \leq h$, for which $S_i \cap H \neq \emptyset$, for all $i = 1, \dots, m$?

Exercise 11. Prove that the following problem is **NP**-complete.

PARTITION

INSTANCE: A set $A = \{a_1, \dots, a_n\}$ of positive integers

QUESTION: Is there a partition of A into two subsets A_1 and A_2 such that

$$\sum_{a \in A_1} a = \sum_{a \in A_2} a.$$

Exercise 12. Prove that the following problem is **NP**-complete.

i-PARTITION

INSTANCE: A set $S = \{s_1, \dots, s_n\}$ of non-negative integers and a positive integer $i \leq n$.

QUESTION: Is there a partition of S into i subsets $S_1, \dots, S_i$ such that

$$\sum_{x \in S_1} x = \sum_{x \in S_2} x = \cdots = \sum_{x \in S_i} x.$$

Exercise 13. Prove that the following problem is **NP**-complete.

SUBGRAPH ISOMORPHISM

INSTANCE: Two graphs G and H.

QUESTION: Is there a subgraph of G which is isomorphic to H?

Exercise 14. Prove that the following problem is **NP**-complete.

LONGEST CYCLE

INSTANCE: A graph G and a bound $b \in \mathbb{Z}^+$.

QUESTION: Is there a cycle in G of length at least b?

Exercise 15 (*)**.** Prove that the following problem is **NP**-complete.

3-DIMENSIONAL MATCHING (3-DM)

INSTANCE: Three disjoint sets A, B, C of size n each. A subset $W \subseteq A \times B \times C$.

QUESTION: Is there $M \subseteq W$ such that each element of $A \cup B \cup C$ appears in M exactly once.

For all problems we have seen so far, the fact that they are in **NP** are often trivial. This is not the case with a few important problems, as the following exercises show.

Exercise 16 (Subtlety of **NP** membership)**.** Consider the following problem (known to be in **NP**):

INTEGER PROGRAMMING

INSTANCE: An $m \times n$ integer matrix A and a vector $b \in \mathbb{Z}^m$.

QUESTION: Is there a vector $x \in \mathbb{Z}^n$ such that $Ax \geq b$?

It is very tempting to say that INTEGER PROGRAMMING is in $\mathbf{NP}$ because, given a certificate vector $y \in \mathbb{Z}^n$ we can verify in polynomial time if $Ay \geq b$. What is wrong with the above reasoning?

Exercise 17 (INTEGER PROGRAMMING is in $\mathbf{NP}$ **)**.** Let A be an $m \times n$ integer matrix and b be a vector in $\mathbb{Z}^m$. Let α be the maximum value of integers in A and b. Let $q = \max\{m, n\}$.

(i) Let B be any square submatrix of A, prove that $|\det(B)| \leq (\alpha q)^q$.

(ii) Prove that, if rank$(A) = r < m$, then there exists a non-zero integer vector z such that $Az = 0$ and the maximum absolute value of integers in z is at most $(\alpha q)^q$.

(iii) Suppose $Ax \geq b$ has an integer solution x. Let a_i denote the ith row of A, b_i the ith component of b, and e_j the jth m-dimensional unit vector. Let x be a solution for $Ax \geq b$ that maximizes the number of elements in the following set:

$$\begin{aligned}\mathcal{A}_x &= \{a_i \mid b_i \leq a_i x \leq b_i + (\alpha q)^{q+1}, i \in [n]\} \\ &\quad \cup \{e_j \mid |x_j| \leq (\alpha q)^q, j \in [m]\}.\end{aligned}$$

Prove that rank$(\mathcal{A}_x) = m$.

(iv) Prove that, if there is an integer vector x satisfying $Ax \geq b$, then there is an integer vector y satisfying $Ay \geq b$ for which absolute values of integers in y are at most $2(\alpha q)^{2q+2}$.

(v) Prove that INTEGER PROGRAMMING is in $\mathbf{NP}$.

Exercise 18 (PRIME is in $\mathbf{NP}$ **)**.** The PRIME problem is the problem of determining whether an input number n is prime or composite. In August of 2002, Agrawal, Kayal, and Saxena of IIT Kanpur showed that PRIME is in $\mathbf{P}$. Their proof is actually accessible to any one with rudimentary knowledge of abstract algebra. (See [1] for an interesting story.)

Prove a much weaker result that PRIME $\in \mathbf{NP}$ using the following known fact: an odd integer $p > 2$ is a prime if and only if there exists an integer $a \in \{1, 2, \ldots, p-1\}$ such that

(i) $a^{p-1} \equiv 1 \pmod{p}$.

(ii) $a^{(p-1)/q} \not\equiv 1 \pmod{p}$ for all prime factors q of $p - 1$.

References

[1] F. BORNEMANN, *PRIMES is in P: a breakthrough for "Everyman"*, Notices Amer. Math. Soc., 50 (2003), pp. 545–552.

[2] D.-Z. DU AND K.-I. KO, *Theory of computational complexity*, Wiley-Interscience Series in Discrete Mathematics and Optimization, Wiley-Interscience, New York, 2000.

[3] M. R. GAREY AND D. S. JOHNSON, *Computers and intractability*, W. H. Freeman and Co., San Francisco, Calif., 1979. A guide to the theory of NP-completeness, A Series of Books in the Mathematical Sciences.