

1 Introduction

So far, students have seen the design of polynomial time (i.e., efficient) algorithms for various problems (e.g., finding a minimum spanning tree of a graph, finding shortest paths between pairs of vertices in a graph). This class of efficiently solvable is usually denoted by **P**. In this and the next lecture, we will study some problems (which belong to the class of **NP**-complete problems) for which no efficient algorithms are known. It is widely believed that there are no efficient algorithms for these problems; however, at present, we cannot prove that no efficient algorithms exist for these problems.

Notes regarding terminology: We will treat the phrases “efficiently solvable” and “polynomial time solvable” as synonyms. Also, when we discuss graph problems, we treat the words “node” and “vertex” as synonyms. When we consider Boolean formulas, the values 1 and 0 denote **True** and **False** respectively.

1.1 Some Basic Definitions

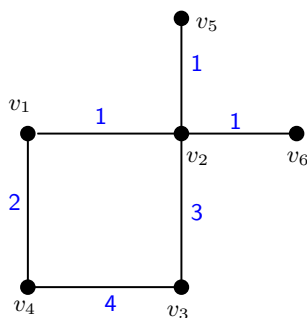
The study of NP-completeness considers **decision problems**, that is, problems where the answer is “YES” or “NO”. (This is because the formal definition of the class is based on language acceptance problems for Turing Machines.) Each problem is specified by an **instance** and a **question** whose answer is “YES” or “NO”. Here are two examples.

Example: Here is a decision version of the Minimum Spanning Tree (MST) problem.

Instance: An undirected graph $G(V, E)$ with a weight $w(e)$ for each edge $e \in E$ and a number B .

Question: Does G have a spanning tree of weight *at most* B ?

Thus, each instance of the MST problem is specified using an edge weighted undirected graph G and a number B . Some are “YES” instances and others are “NO” instances. Examples of a “YES” instance and a “NO” instance for the MST problem are shown below.



- For the graph G (on the left), if we set $B = 8$, we get a “YES” instance of the MST problem.
- For the graph G , if we set $B = 7$, we get a “NO” instance of the MST problem.

Before discussing the next example (namely, the Boolean Satisfiability problem), we recall a few standard definitions from Boolean algebra.

- For a Boolean variable x (which takes a value from $\{0, 1\}$), the **complement** of x is denoted by \bar{x} . Each of x and \bar{x} is called a **literal**.
- A **clause** is a disjunction (i.e., the OR) of literals. For instance, if x_1, x_2 and x_3 are Boolean variables, examples of clauses are (\bar{x}_2) , $(x_1 \vee \bar{x}_3)$ and $(x_1 \vee \bar{x}_2 \vee \bar{x}_3)$.
- When we assign a 0 or 1 value to each Boolean variable, a clause evaluates to 1 (i.e., it is **satisfied**) or 0 (it is not satisfied).

Example: The Boolean **Satisfiability** (SAT) problem can be defined as follows.

Instance: A set $X = \{x_1, x_2, \dots, x_n\}$ of Boolean variables and a set $F = \{C_1, C_2, \dots, C_m\}$ of m clauses formed using the variables in X . (**Note:** Think of F as the Boolean formula $C_1 \wedge C_2 \wedge \dots \wedge C_m$. Such a formula is said to be in **Conjunctive Normal Form** or CNF.)

Question: Is F satisfiable (i.e., is there an assignment of 0-1 values to each variable in X such that each clause in F is satisfied)?

Examples of “YES” and “NO” instances of SAT are given below.

- Let $X = \{x_1, x_2, x_3, x_4\}$. Suppose F_1 consists of two clauses: $C_1 = (x_1 \vee \bar{x}_3 \vee x_4)$ and $C_2 = (\bar{x}_2 \vee x_3 \vee \bar{x}_4)$. This is a “YES” instance of SAT since by setting $x_1 = 1, x_2 = 0, x_3 = 0$ and $x_4 = 0$, both the clauses can be satisfied.
- Let $X = \{x_1, x_2\}$. Suppose F_2 consists of three clauses: $C_1 = (x_1 \vee x_2)$, $C_2 = (\bar{x}_1)$ and $C_3 = (\bar{x}_2)$. This is a “NO” instance of SAT for the following reason. To satisfy C_2 and C_3 , we must set $x_1 = x_2 = 0$. However, this choice does not satisfy C_1 .

1.2 Complexity Classes P and NP

As mentioned earlier, the class **P** contains problems that can be solved efficiently (i.e., in time polynomial in the size of the problem instance). MST is an example of such a problem.

The class **NP** (Nondeterministic Polynomial time) is defined as the set of problems for which a *given solution can be verified efficiently*. This requires two conditions:

- (i) the size of the given solution S should itself be a polynomial in the size of the problem instance I and
- (ii) the verification algorithm must run in time that is a polynomial in the sum of the sizes of the problem instance I and the solution S .

The given solution S is also called a **certificate**.

Example – SAT Problem: We can argue that SAT is in **NP** as follows.

- Suppose a given instance I of SAT has n variables and m clauses. We can consider the size of the instance I as $O(mn)$.
- A proposed solution (certificate) $S = (b_1, b_2, \dots, b_n)$, where $b_i \in \{0, 1\}$, $1 \leq i \leq n$, is a vector with n Boolean values, with b_i giving the value for variable x_i . Thus, the size of the proposed solution S is $O(n)$, which is a polynomial function of the size of the SAT instance I .
- Given S , we can substitute the values into each clause and check whether a clause is satisfied in $O(n)$ time. So, the total time to check that all the clause are satisfied is $O(mn)$, which is a polynomial in the sum of the sizes of I and S .

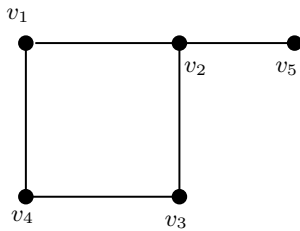
Thus, SAT is in **NP**.

Example – Minimum Vertex Cover Problem: The Minimum Vertex Cover (MVC) problem is defined as follows.

Instance: An undirected graph $G(V, E)$ and an integer $k \leq |V|$.

Question: Does G have a **vertex cover** of size at most k , that is, is there a subset $V' \subseteq V$ such that $|V'| \leq k$ and for each edge $\{v_i, v_j\} \in E$, at least one of v_i and v_j is in V' ?

An example to illustrate the notion of vertex cover is given below.



- For the graph G (on the left), $V_1 = \{v_2, v_4\}$ is a vertex cover. (It is also a minimum vertex cover.)
 - For the graph G , $V_2 = \{v_1, v_3\}$ is not a vertex cover. (Edge $\{v_2, v_5\}$ is not covered.)
-

We can argue that MVC is **NP** as follows.

- A certificate for MVC is a subset of vertices V' . Thus, the size of the certificate is polynomial in the size of the graph G .
- It is easy to see that given G and V' , one can verify in polynomial time that $|V'| \leq k$ and that for each edge $\{v_i, v_j\} \in E$, at least one of v_i and v_j is in V' .

Thus, MVC is in **NP**.

Note: One can also think of a nondeterministic machine as having the capability to “guess” a solution (i.e., a certificate) and then deterministically verify that the certificate is indeed a solution.

Relationship between P and NP: It can be seen that $\mathbf{P} \subseteq \mathbf{NP}$. The reason is that for any problem in **P**, a polynomial time algorithm can construct a solution and use the solution as the certificate. (For example, for the MST problem, an algorithm can construct a minimum spanning tree T for the given graph and use T as the certificate to verify that there is a spanning tree of cost at most B .)

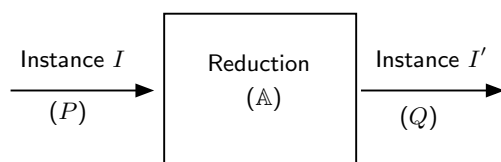
However, whether $\mathbf{P} = \mathbf{NP}$ is a famous open question. (It is one of the problems for which Clay Institute has offered a prize of 1 Million US dollars.) It is widely believed that $\mathbf{P} \neq \mathbf{NP}$.

1.3 Polynomial Time Reductions

A **reduction** from a problem P to a problem Q is a deterministic algorithm \mathbb{A} which transforms any instance I of problem P to an instance I' of problem Q such that

- (i) \mathbb{A} runs in polynomial time and
- (ii) I is a “YES” instance of P if and only if I' is a “YES” instance of Q .

If such a reduction exists, we say that “ P is reducible to Q ”; formally, this is denoted by $P \leq_p Q$. (The subscript ‘ p ’ is used to indicate that the reduction runs in polynomial time.)



Thus, a reduction \mathbb{A} efficiently transforms each instance I of P into an instance I' of Q such that there is a solution to I iff there is a solution to I' .

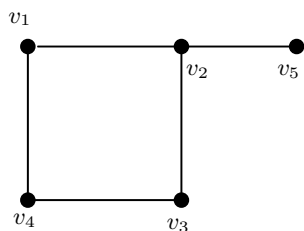
1.4 Examples of Reductions

(a) Reduction from Minimum Vertex Cover (MVC) to Maximum Independent Set: We have seen the definition of the MVC problem. The definition of the Maximum Independent Set (MIS) problem is as follows.

Instance: An undirected graph $G(V, E)$ and an integer $\ell \leq |V|$.

Question: Does G have an **independent set** with *at least* ℓ vertices, that is, is there a subset $V_1 \subseteq V$ such that $|V_1| \geq \ell$ and there is no edge in G between any pair of vertices in V_1 ?

An example to illustrate the notion of independent set is given below.



- For the graph G (on the left), $V_1 = \{v_1, v_3, v_5\}$ is an independent set. (It is also a maximum independent set.)
- For the graph G , $V_2 = \{v_1, v_4, v_5\}$ is not an independent set because G has the edge $\{v_1, v_4\}$.

We now show that there is a polynomial time reduction from MVC to MIS. (This reduction is discussed in [3].)

Theorem 1.1 $MVC \leq_p MIS$.

Proof: Let the given instance I of the MVC problem consist of graph $G(V, E)$ and integer k . The reduction must efficiently construct an instance I' of the MIS problem such that there is a solution to the MVC instance I iff there is a solution to the MIS instance I' . (Note that I' consists of a graph $G'(V', E')$ and an integer ℓ .)

1. The graph $G'(V', E')$ is the same as $G(V, E)$.

2. Let $\ell = |V| - k$.

This completes the reduction. Note that the time needed to carry out the reduction is $O(|V| + |E|)$, which is polynomial in the size of instance I . We must now prove that there is a solution to the MVC instance I iff there is a solution to the MIS instance I' . We do this two parts as follows.

Part 1: Suppose there is a solution V_1 to the MIS instance I' . Thus, $|V_1| \geq \ell$. We will show that $V - V_1$ is a solution to the MVC instance I (i.e., $V - V_1$ is a vertex cover for G and $|V - V_1| \leq k$). To show that $V - V_1$ is a vertex cover, consider any edge $\{v_i, v_j\} \in G$. Since V_1 is an independent set, both v_i and v_j *cannot* be in V_1 . In other words, at least one of v_i and v_j is in $V - V_1$; that is, $V - V_1$ is indeed a vertex cover for G . Also note that $|V - V_1| \leq |V| - \ell = k$ (since $\ell = |V| - k$). Thus, $V - V_1$ is indeed a solution to the MVC instance I .

Part 2: Suppose there is a solution V' to the MVC instance I . We will show that $V - V'$ is a solution to the MIS instance I' . To see that $V - V'$ is an independent set, consider any pair of vertices v_i and v_j in $V - V'$. If G has the edge $\{v_i, v_j\}$, then V' cannot be vertex cover (since neither v_i nor v_j is in V'). Therefore, there cannot be an edge between any pair of vertices in $V - V'$; thus, $V - V'$ is an independent set. Also, $|V - V'| \geq |V| - k = \ell$. Hence, $V - V'$ is a solution to the MIS instance I' .

This completes the proof that $\text{MVC} \leq_p \text{MIS}$. ■

(b) Reduction from Graph 2-Coloring to 2SAT: We will consider the following two problems.

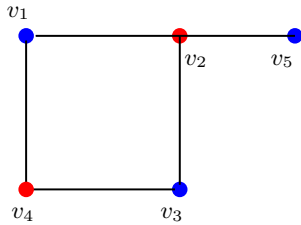
(i) 2SAT: This the SAT problem in which each clause contains at most two literals. (This problem is known to be efficiently solvable; see for example [5].)

(ii) Graph 2-Coloring (G2C): This decision problem is defined as follows.

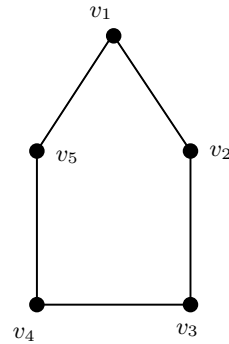
Instance: Undirected graph $G(V, E)$.

Question: Can each vertex in V be assigned a color from $\{\text{Red}, \text{Blue}\}$, so that for each edge $\{v_i, v_j\} \in E$, the colors assigned to v_i and v_j are different?

An example to illustrate the Graph 2-Coloring problem is given below.



(A 2-colorable graph)



(This graph is not 2-colorable.)

We now show that G2C can be efficiently reduced to 2SAT.

Theorem 1.2 $G2C \leq_p 2SAT$.

Proof: The intuitive idea behind the proof is the following.

- Each vertex of the graph in G2C becomes a Boolean variable in 2SAT.
- Any vertex corresponding to a Boolean variable assigned the value 1 (0) in 2SAT is colored Blue (Red).
- The condition that the end points of an edge must be colored differently is enforced by creating appropriate clauses in 2SAT.

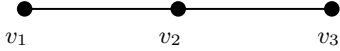
Let the G2C instance I be given by the graph $G(V, E)$, where $V = \{v_1, v_2, \dots, v_n\}$. We create an instance I' of 2SAT as follows.

1. For each vertex v_i , create a Boolean variable x_i , $1 \leq i \leq n$.
2. For each edge $\{v_i, v_j\}$ in G , create the following two clauses: $(x_i \vee x_j)$, $(\overline{x_i} \vee \overline{x_j})$

Note that each resulting clause has exactly two literals. Thus, I' is an instance of 2SAT.

The reduction can be done in polynomial time since the set of variables can be constructed in $O(|V|)$ time and the set of clauses can be constructed in $O(|E|)$ time.

An example to illustrate this construction is shown below.

	Boolean variables: x_1, x_2, x_3
	Clauses: $(x_1 \vee x_2), (\overline{x_1} \vee \overline{x_2}), (x_2 \vee x_3), (\overline{x_2} \vee \overline{x_3})$

Why did we choose these clauses? When G has the edge $\{v_i, v_j\}$, we want v_i and v_j to have different colors. Since the two colors correspond to Boolean values, we want the two corresponding variables x_i and x_j to have *different* Boolean values. This is achieved by the two chosen clauses. If both x_i and x_j are 1, then the clause $(\overline{x_i} \vee \overline{x_j})$ won't be satisfied. Likewise, if both x_i and x_j are 0, then the clause $(x_i \vee x_j)$ won't be satisfied. Thus, in any satisfying assignment, x_i and x_j must have different Boolean values.

We will now show that the G2C instance I has a solution iff the 2SAT instance I' has a solution.

Part 1: Suppose the 2SAT instance I' has a solution. We will construct a solution to the G2C instance I as follows. For $1 \leq i \leq n$, if the variable x_i is set to 1, we choose the color of the corresponding vertex v_i to Blue; otherwise, we choose the color of v_i to be Red. To prove that this is a valid 2-coloring of G , consider any edge $\{v_i, v_j\}$. As discussed above, the two clauses of 2SAT corresponding to this edge ensure that the variables x_i and x_j have different values. Thus, in the constructed solution to G2C, v_i and v_j have different colors.

Part 2: Suppose the G2C instance I has a solution. We construct a solution to the 2SAT instance I' as follows. For $1 \leq i \leq n$, if vertex v_i is colored Blue, we set the variable x_i to 1; otherwise, we set x_i to 0. We will prove that this satisfies all the clauses of the 2SAT instance I' . Consider any clause, say C . From our construction, C is of the form either $(x_i \vee x_j)$ or $(\overline{x_i} \vee \overline{x_j})$ for some variables x_i and x_j . Since the two corresponding vertices v_i and v_j have different colors, x_i and x_j have different values. Thus, in both forms of clauses, one of the literals has the value 1; that is, the clause is satisfied.

This completes the proof that $G2C \leq_p 2SAT$. ■

Usefulness of Reductions: Polynomial time reductions are useful in both obtaining efficient algorithms to new problems as well as showing that problems are computationally intractable. The following lemma expresses this idea formally.

Lemma 1.1 *Suppose $P \leq_p Q$.*

- (i) *If Q can be solved in polynomial time, then P can also be solved in polynomial time.*
- (ii) *If P cannot be solved in polynomial time, then Q cannot be solved in polynomial time.*

Proof: For Part (i), notice that a polynomial time algorithm for P can be constructed as follows:

1. Given an instance I of P , use the polynomial time reduction to construct an instance I' of Q .
2. Solve the instance I' using the known algorithm for Q . The answer to I is “YES” iff the answer to I' is “YES”.

Part (ii) can be shown easily by contradiction. If Q can be solved in polynomial time, then the two step procedure mentioned above will provide a polynomial time algorithm for P , and this contradicts the assumption that P cannot be solved in polynomial time. ■

Notes:

1. 2SAT is known to be efficiently solvable [5]. This the reduction from Graph 2-Coloring (G2C) to 2SAT presented above points out that G2C can be solved efficiently.
2. The Minimum Vertex Cover (MVC) problem is known to **NP**-complete. As we will see, the reduction from MVC to Maximum Independent Set (MIS) discussed above can be used to show that MIS is also **NP**-complete.

Thus, reductions can be used to prove that a certain problem is “easy” (i.e., efficiently solvable) and that another problem is “hard” (e.g., **NP**-complete).

1.5 What we know about NP-complete Problems

Informally, **NP**-complete problems are the “hardest” ones in **NP**. Section B of the appendix defines this idea more formally.

All the **NP**-complete problems are equivalent in the following sense. Either they are efficiently solvable or none of them has an efficient algorithm. Unfortunately, at this time, we don’t know which of these two possibilities is true. It is widely believed that **NP**-complete problems cannot be solved efficiently. The best known algorithms for **NP**-complete problems have running times that are exponential in the size of the problem. So, such problems are referred to “computationally intractable” problems.

1.6 Steps Used to Prove NP-completeness

Suppose we want to prove the **NP**-completeness of a problem Y . The sequence of steps to be used in proving such a result is as follows.

1. Show that problem Y is in **NP**. (For this, you need to indicate how to specify a solution (certificate) and verify in polynomial time that the solution satisfies all the required properties.)
2. Identify a suitable problem X which is known to be **NP**-complete.
3. Develop an algorithm that transforms any instance I_x of problem X into an instance I_y of problem Y .
4. Show that the transformation described in Step 3 can be carried out in polynomial time.
5. Show that there is a solution to instance I_x of X if and only if there is a solution to instance I_y of Y .

Step 1 above proves the membership of problem Y in **NP**. The remaining steps prove the **NP**-hardness of problem Y .

We now briefly discuss how the above steps can be used to show that the MIS problem is **NP**-complete using the fact that the MVC problem is **NP**-complete.

- MIS is in **NP** since a proposed solution (certificate) is a subset V' of vertices. It is easy to see that checking whether $|V'| \geq \ell$ and that V' forms an independent set can be done efficiently. (This is Step 1 above.)
- Starting from the known **NP**-complete problem, we presented a polynomial time reduction in the proof of Theorem 1.1. That proof carries out Steps 2 through 5 mentioned above.

Hence, we can conclude that the MIS problem is **NP**-complete.

1.7 Important Notes About Reductions

- To show that a problem P is “easy” (i.e., efficiently solvable) through a reduction, you must first identify a problem Q that is known to be easy and construct a polynomial time reduction from P to Q (i.e., show that $P \leq_p Q$). Thus, in this case, P should be the starting point (or source) of the reduction.
- To show that a problem P is “hard” (i.e., **NP**-complete), you must first identify a problem Q that is known to be **NP**-complete and construct a polynomial time reduction from Q to P (i.e., show that $Q \leq_p P$). Thus, in this case, P should be the end point (or target) of the reduction.

2 Suggested Exercises

1. **[Easy]** The Maximum Spanning Tree (MxST) problem is similar to the Minimum Spanning Tree (MST) problem, except that the goal is to find a spanning tree of *maximum* total weight. Show that $\text{MxST} \leq_p \text{MST}$. (This reduction shows that MxST can also be solved efficiently.)

2. **[Easy]** The Maximum Clique (MxC) problem is defined as follows.

Instance: An undirected graph $G(V, E)$ and an integer $k \leq |V|$.

Question: Does G have a clique with *at least* ℓ vertices, that is, is there a subset $V_1 \subseteq V$ such that $|V_1| \geq \ell$ and there is an edge in G between every pair of vertices in V_1 ?

Show that $\text{MIS} \leq_p \text{MxC}$. (It is easy to see that MxC is in **NP**. Since MIS is **NP**-complete, this reduction shows that MxC is also **NP**-complete.)

3. Let 3SAT denote the version of SAT where each clause contains exactly 3 literals. Note that each solution to 3SAT is an assignment of values to the Boolean variables so that *at least one* literal in each clause has the value 1. Consider the following version of 3SAT called Strong 3SAT (S3SAT):

Instance: A set $X = \{x_1, x_2, \dots, x_n\}$ of Boolean variables and set a set $F = \{C_1, C_2, \dots, C_m\}$ of m clauses formed using the variables in X such that each clause has exactly 3 literals.

Question: Is there an assignment of 0-1 values to the variables in X such that each clause in F has *at least two* literals that are set to 1?

Show that $\text{S3SAT} \leq_p \text{2SAT}$. (Since 2SAT is efficiently solvable, this reduction shows that S3SAT is also efficiently solvable.)

4. Let $G(V, E)$ be a complete undirected graph (i.e., there is an edge between each pair of vertices in V) such that for each edge $\{u, v\}$, there is a positive distance value $d(u, v)$. The **diameter** of G is the largest value among all the edge distances in G . (If G contains only one node, the diameter of G is zero.) When we partition V into two subsets V_1 and V_2 , we get two complete subgraphs $G_1(V_1, E_1)$ and $G_2(V_2, E_2)$, where E_1 and E_2 are respectively the set of edges in the complete graph on V_1 and V_2 respectively. Thus, the diameters of G_1 and G_2 are well defined. Consider the following problem called Two-Clustering (TC):

Instance: A complete undirected graph $G(V, E)$ with a positive distance value $d(u, v)$ for each edge $\{u, v\}$ and positive numbers α_1 and α_2 , where $\alpha_1 \geq \alpha_2$.

Question: Is there a partition of the node set V into two subsets V_1 and V_2 such that the diameter of the subgraph $G_1(V_1, E_1)$ is $\leq \alpha_1$ and the diameter of the subgraph $G_2(V_2, E_2)$ is $\leq \alpha_2$?

Show that $\text{TC} \leq_p \text{2SAT}$. (This reduction shows that TC is efficiently solvable.)

References

- [1] S. A. Cook. The complexity of theorem-proving procedures. In *Proc. Third Annual ACM symposium on Theory of computing (STOC)*, pages 151–158, 1971.
- [2] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness*. W. H. Freeman & Co., San Francisco, 1979.
- [3] J. Kleinberg and E. Tardos. *Algorithm Design*. Addison Wesley, Reading, MA, 2006.
- [4] L. A. Levin. Universal sequential search problems. *Problemy peredachi informatsii*, 9(3):115–116, 1973.
- [5] C. H. Papadimitriou. *Computational Complexity*. Addison Wesley, Reading, MA, 1994.

A Glossary: Definitions of Some NP-Complete Problems

We give below the definitions of some well known **NP**-complete problems. A much longer list can be found in [2].

(a) Boolean Satisfiability (SAT)

Instance: A set $X = \{x_1, x_2, \dots, x_n\}$ of Boolean variables and set a set $F = \{C_1, C_2, \dots, C_m\}$ of m clauses formed using the variables in X . (**Note:** Think of F as the Boolean formula $C_1 \wedge C_2 \wedge \dots \wedge C_m$.)

Question: Is F satisfiable (i.e., is there an assignment of 0-1 values to the variables in X such that each clause in F is satisfied)?

Notes: The SAT problem is **NP**-complete even when each clause contains exactly 3 literals. This version is called 3SAT. If each clause contains at most 2 literals, the corresponding problem is called 2SAT which can be solved efficiently.

(b) Minimum Vertex Cover (MVC)

Instance: An undirected graph $G(V, E)$ and an integer $k \leq |V|$.

Question: Does G have a vertex cover of size at most k , that is, is there a subset $V' \subseteq V$ such that $|V'| \leq k$ and for each edge $\{v_i, v_j\} \in E$, at least one of v_i and v_j is in V' ?

(c) Maximum Independent Set (MIS)

Instance: An undirected graph $G(V, E)$ and an integer $\ell \leq |V|$.

Question: Does G have an independent set with *at least* ℓ vertices, that is, is there a subset $V_1 \subseteq V$ such that $|V_1| \geq \ell$ and there is no edge in G between any pair of vertices in V_1 ?

(d) Graph k-Coloring (GC)

Instance: Undirected graph $G(V, E)$ and an integer $k \leq |V|$.

Question: Can each vertex in V be assigned a color from $\{1, 2, \dots, k\}$, so that for each edge $\{v_i, v_j\} \in E$, the colors assigned to v_i and v_j are different?

Notes: For $k = 2$, we have the Graph 2-Coloring (G2C) problem which can be solved efficiently. For any constant $k \geq 3$, the GC problem remains **NP**-complete.

(e) Minimum Set Cover (MSC)

Instance: A universal set $U = \{u_1, u_2, \dots, u_n\}$, a collection $S = \{S_1, S_2, \dots, S_m\}$, where each S_j is a subset of U ($1 \leq j \leq m$), and an integer $r \leq m$.

Question: Is there is a subcollection S' of S such that $|S'| \leq r$ and the union of the sets in S' is equal to U ?

(f) Exact Cover (XC)

Instance: A universal set $U = \{u_1, u_2, \dots, u_n\}$, a collection $S = \{S_1, S_2, \dots, S_m\}$, where each S_j is a subset of U ($1 \leq j \leq m$).

Question: Is there is a subcollection S' of S such that the sets in S' are pairwise disjoint and the union of the sets in S' is equal to U ?

Note: The XC problem remains **NP**-complete even when $|U|$ is an integer multiple of 3 and each set in S has exactly 3 elements. This restricted version is called Exact Cover by 3-Sets (X3C).

(g) Minimum Dominating Set (MDS)

Instance: An undirected graph $G(V, E)$ and an integer $k \leq |V|$.

Question: Does G have a dominating set of size at most k , that is, is there a subset $V' \subseteq V$ such that $|V'| \leq k$ and for each vertex $v_i \in V - V'$, there is some vertex $v_j \in V'$ such that $\{v_i, v_j\} \in E$?

B Formal Definition of NP-completeness

This section is based on the material presented in Chapter 2 of the classic text on **NP**-completeness by Michael Garey and David Johnson [2]. We begin with a simple fact about polynomial time reductions.

Fact B.1 *The binary relation \leq_p is transitive; that is, if $P \leq_p Q$ and $Q \leq_p R$, then $P \leq_p R$.*

A formal definition of an **NP**-complete problem is as follows.

Definition B.1 *A problem P is **NP-complete** if (i) P is in **NP** and (ii) for every problem R in **NP**, $R \leq_p P$.*

It is not possible to use this definition directly to prove **NP**-completeness results since it requires that we need to show polynomial time reductions from every problem in **NP**. The following lemma points out that things would become simpler if we can identify one **NP**-complete problem.

Lemma B.1 *If P and Q are in **NP**, P is **NP-complete** and $P \leq_p Q$, then Q is also **NP-complete**.*

Proof: Since Q is in **NP**, we only need to show That for every problem R in **NP**, $R \leq_p Q$. Consider any problem R in **NP**. Since P is **NP-complete**, we know that $R \leq_p P$. We also know that $P \leq_p Q$. By transitivity of reductions, $R \leq_p Q$. Thus, Q is **NP-complete**. ■

So, we need some **NP**-complete problem to start with. The honor of being the “first” **NP**-complete problem belongs to the Satisfiability (SAT) problem. The **NP**-completeness of SAT was established by Steve Cook (University of Toronto) in 1971 [1]. The result was also established independently [4] by Leonid Levin (Boston University) who was then in the Soviet Union. We now know of many **NP**-complete problems and we can use any of them as the starting problem to establish **NP**-completeness results.