Introduction to NP-Completeness: Lecture 18

Lecturer: S. S. Ravi

Affiliation: Biocomplexity Institute and Initiative, UVA

Email: ssravi@virginia.edu

Date: Oct. 27, 2020

Request: If you send email to Ravi regarding this lecture, please be sure to cc the message to Professor Haifeng Xu (hx4ad@virginia.edu).

Outline for Lecture 18

1. Basic Definitions (Decision Problems and Examples)
2. Complexity Classes **P** and **NP**
3. Polynomial Time Reductions with Examples

Basic Definitions

Terminology and Notation:

- "Efficiently solvable" and "Polynomial time solvable" are synonyms.
- For graph problems, "vertex" and "node" are synonyms.
- The Boolean values `True` and `False` are represented by 1 and 0 respectively.

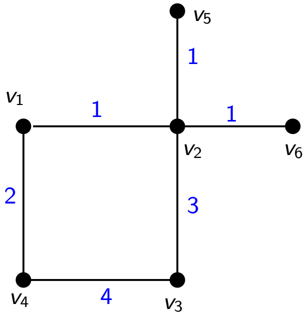
Decision Problem:

- Problem specification consists of an **instance** and **question**.
- The answer to the question is "YES" or "NO".

Example – Minimum Spanning Tree (MST):

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?



- For this graph G, choosing $B = 8$ leads to a "YES" instance.
- For the same graph, choosing $B = 7$ leads to a "NO" instance.

Basic Definitions (continued)

Additional Terminology:

- For a Boolean variable x, its **complement** is $\overline{x}$; x and $\overline{x}$ are **literals**.
- A **clause** is a disjunction (i.e., the OR) of literals.

  Examples: $(\overline{x_2})$, $(x_1 \vee \overline{x_3})$, $(x_1 \vee \overline{x_2} \vee \overline{x_3})$
- 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).

Satisfiability (SAT) Problem:

Instance: A set $X = \{x_1, x_2, \ldots, x_n\}$ of Boolean variables and a set $F = \{C_1, C_2, \ldots, C_m\}$ of m clauses using the variables in X.

Note: Think of F as the formula $C_1 \wedge C_2 \wedge \cdots \wedge C_m$. Such a formula is in **Conjunctive Normal Form** (CNF).

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

Examples:

1. Let $X = \{x_1, x_2, x_3, x_4\}$ and let F_1 consist of $C_1 = (x_1 \vee \overline{x_3} \vee x_4)$ and $C_2 = (\overline{x_2} \vee x_3 \vee \overline{x_4})$. This is a "YES" instance of SAT. (Choose $x_1 = 1$, $x_2 = 0$, $x_3 = 0$ and $x_4 = 0$.)

Basic Definitions (continued)

Satisfiability (SAT) Problem:

Instance: A set $X = \{x_1, x_2, \ldots, x_n\}$ of Boolean variables and a set $F = \{C_1, C_2, \ldots, C_m\}$ of m clauses using the variables in X.

Note: Think of F as the formula $C_1 \wedge C_2 \wedge \cdots \wedge C_m$. Such a formula is in **Conjunctive Normal Form** (CNF).

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

Examples:

1. Let $X = \{x_1, x_2, x_3, x_4\}$ and let F_1 consist of $C_1 = (x_1 \vee \overline{x_3} \vee x_4)$ and $C_2 = (\overline{x_2} \vee x_3 \vee \overline{x_4})$. This is a "YES" instance of SAT. (Choose $x_1 = 1$, $x_2 = 0$, $x_3 = 0$ and $x_4 = 0$.)

2. Let $X = \{x_1, x_2\}$. Suppose F_2 consists of $C_1 = (x_1 \vee x_2)$, $C_2 = (\overline{x_1})$ and $C_3 = (\overline{x_2})$. This is a "NO" instance of SAT. **(Why?)**

Complexity Classes **P** and **NP**

Class P: Contains problems that can be solved in polynomial time. (MST is an example of such a problem.)

Class NP (Nondeterministic Polynomial time): Contains problems for which a *given solution can be verified efficiently*.

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

This verification requires two conditions.

1. The size of the given solution S should be a polynomial in the size of the problem instance I.

Complexity Classes **P** and **NP**

Class P: Contains problems that can be solved in polynomial time. (MST is an example of such a problem.)

Class NP (Nondeterministic Polynomial time): Contains problems for which a *given solution can be verified efficiently*.

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

This verification requires two conditions.

1. The size of the given solution S should be a polynomial in the size of the problem instance I.
2. 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.

Complexity Classes **P** and **NP** (continued)

Example – SAT is in NP:

- Consider a given instance I of SAT with n variables and m clauses. (The size of the instance I can be taken as $O(mn)$.)

Complexity Classes **P** and **NP** (continued)

Example – SAT is in NP:

- Consider a given instance I of SAT with n variables and m clauses. (The size of the instance I can be taken as $O(mn)$.)
- A proposed solution (certificate) is:
$$S = (b_1, b_2, \ldots, b_n),$$
where $b_i \in \{0, 1\}$, is the value of x_i, $1 \leq i \leq n$. (The size of this certificate is $O(n)$.)

Complexity Classes **P** and **NP** (continued)

Example – SAT is in NP:

- Consider a given instance I of SAT with n variables and m clauses. (The size of the instance I can be taken as $O(mn)$.)
- A proposed solution (certificate) is:

$$S = (b_1, b_2, \ldots, b_n),$$

  where $b_i \in \{0, 1\}$, is the value of x_i, $1 \leq i \leq n$. (The size of this certificate is $O(n)$.)
- Given S, we can substitute the values into each clause check that all clauses are satisfied in $O(mn)$ time.

Complexity Classes **P** and **NP** (continued)

Example – SAT is in NP:

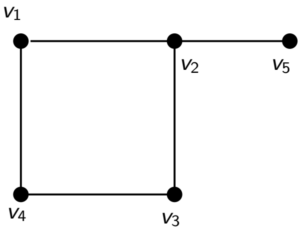
- Consider a given instance I of SAT with n variables and m clauses. (The size of the instance I can be taken as $O(mn)$.)
- A proposed solution (certificate) is:
$$S = (b_1, b_2, \ldots, b_n),$$
where $b_i \in \{0, 1\}$, is the value of x_i, $1 \leq i \leq n$. (The size of this certificate is $O(n)$.)
- Given S, we can substitute the values into each clause check that all clauses are satisfied in $O(mn)$ time.
- Thus, SAT is in **NP**.

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'?

Example:



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

Complexity Classes **P** and **NP** (continued)

Example – Minimum Vertex Cover (MVC) is in NP:

- A given instance I of MVC has a graph $G(V, E)$, with $|V| = n$, $|E| = m$, and an integer $k \leq n$. (The size of instance I is $O(m + n)$.)

Example – Minimum Vertex Cover (MVC) is in NP:

- A given instance I of MVC has a graph $G(V, E)$, with $|V| = n$, $|E| = m$, and an integer $k \leq n$. (The size of instance I is $O(m + n)$.)
- A proposed solution (certificate) is $V' \subseteq V$. (Size of the certificate is $O(n)$.)

Complexity Classes **P** and **NP** (continued)

Example – Minimum Vertex Cover (MVC) is in NP:

- A given instance I of MVC has a graph $G(V, E)$, with $|V| = n$, $|E| = m$, and an integer $k \leq n$. (The size of instance I is $O(m + n)$.)
- A proposed solution (certificate) is $V' \subseteq V$. (Size of the certificate is $O(n)$.)
- Given V', one can efficiently check 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'.

  (**Exercise:** Explain how the verification can be done in $O(m + n)$ time.)

Example – Minimum Vertex Cover (MVC) is in NP:

- A given instance I of MVC has a graph $G(V, E)$, with $|V| = n$, $|E| = m$, and an integer $k \leq n$. (The size of instance I is $O(m + n)$.)
- A proposed solution (certificate) is $V' \subseteq V$. (Size of the certificate is $O(n)$.)
- Given V', one can efficiently check 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'.

  (**Exercise:** Explain how the verification can be done in $O(m + n)$ time.)
- Thus, MVC is in **NP**.

Relationship between **P** and **NP**

Observation: $\mathbf{P} \subseteq \mathbf{NP}$.

Reason: For any problem in **P**, a polynomial time algorithm can construct a solution and use the solution as the certificate.

Example: For the MST problem, an algorithm can construct a minimum spanning tree T and use T as the certificate.

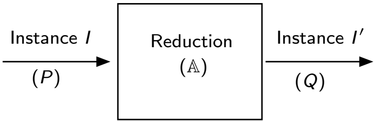
Famous Open Question: Is $\mathbf{P} = \mathbf{NP}$?

- Considered a very important problem in Computer Science and Mathematics.
- Clay Institute offers a prize of $1 Million for its solution.
- Most CS researchers believe that $\mathbf{P} \neq \mathbf{NP}$.

Polynomial Time Reductions

Definition: 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 B such that

1. $\mathbb{A}$ runs in polynomial time and
2. I is a "YES" instance of P if and only if I' is a "YES" instance of Q.



Note: 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'.

Terminology/Notation: "P is reducible to Q" or $P \leq_p Q$.

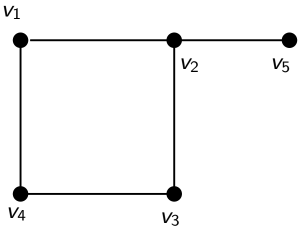
Example – Reducing MVC to Maximum Independent Set:

Maximum Independent Set (MIS):

Instance: An undirected graph $G(V, E)$ and 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?

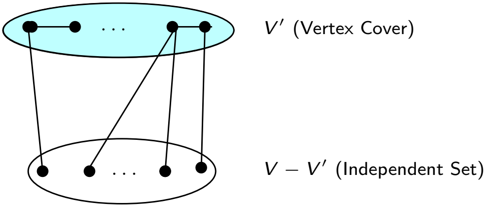
Example:



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

Lemma 1: For any graph $G(V, E)$, V' is a vertex cover iff $V - V'$ is an independent set.

Proof idea:



Corollary 1: $G(V, E)$ has a vertex cover of size k iff it has an independent set of size $|V| - k$.

Theorem 1: MVC $\leq_p$ MIS.

Proof:

- MVC instance I consists of graph $G(V, E)$ and integer k.

Polynomial Time Reductions (continued)

Theorem 1: MVC $\leq_p$ MIS.

Proof:

- MVC instance I consists of graph $G(V, E)$ and integer k.
- MIS instance I' consists of graph $G'(V', E')$ and integer ℓ.

Theorem 1: MVC $\leq_p$ MIS.

Proof:

- MVC instance I consists of graph $G(V, E)$ and integer k.
- MIS instance I' consists of graph $G'(V', E')$ and integer ℓ.
- **Steps of the Reduction:**

Polynomial Time Reductions (continued)

Theorem 1: MVC $\leq_p$ MIS.

Proof:

- MVC instance I consists of graph $G(V, E)$ and integer k.
- MIS instance I' consists of graph $G'(V', E')$ and integer ℓ.
- **Steps of the Reduction:**
  1. $G'(V', E')$ is the same as $G(V, E)$.

Theorem 1: MVC $\leq_p$ MIS.

Proof:

- MVC instance I consists of graph $G(V, E)$ and integer k.
- MIS instance I' consists of graph $G'(V', E')$ and integer ℓ.
- **Steps of the Reduction:**
  1. $G'(V', E')$ is the same as $G(V, E)$.
  2. $\ell = |V| - k$.

Polynomial Time Reductions (continued)

Theorem 1: MVC $\leq_p$ MIS.

Proof:

- MVC instance I consists of graph $G(V, E)$ and integer k.
- MIS instance I' consists of graph $G'(V', E')$ and integer ℓ.
- **Steps of the Reduction:**
  1. $G'(V', E')$ is the same as $G(V, E)$.
  2. $\ell = |V| - k$.
- Time for reduction: $O(|V| + |E|)$.

Polynomial Time Reductions (continued)

Theorem 1: MVC $\leq_p$ MIS.

Proof:

- MVC instance I consists of graph $G(V, E)$ and integer k.
- MIS instance I' consists of graph $G'(V', E')$ and integer ℓ.
- **Steps of the Reduction:**
  1. $G'(V', E')$ is the same as $G(V, E)$.
  2. $\ell = |V| - k$.
- Time for reduction: $O(|V| + |E|)$.
- From Corollary 1, Instance I has a solution iff I' has a solution.

Polynomial Time Reductions (continued)

Theorem 1: MVC $\leq_p$ MIS.

Proof:

- MVC instance I consists of graph $G(V, E)$ and integer k.
- MIS instance I' consists of graph $G'(V', E')$ and integer ℓ.
- **Steps of the Reduction:**
  1. $G'(V', E')$ is the same as $G(V, E)$.
  2. $\ell = |V| - k$.
- Time for reduction: $O(|V| + |E|)$.
- From Corollary 1, Instance I has a solution iff I' has a solution.
- Thus, MVC $\leq_p$ MIS.

Polynomial Time Reductions (continued)

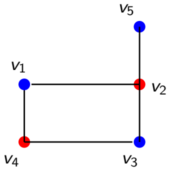
2SAT: Version of SAT in which each clause has at most 2 literals. 2SAT is efficiently solvable.

Graph 2-Coloring: (G2C)

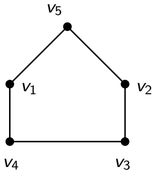
Instance: Undirected graph $G(V, E)$.

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

Example:



(2-Colorable)



(Not 2-Colorable)

Theorem 2: G2C $\leq_p$ 2SAT.

Proof:

- **Intuitive idea:**

Theorem 2: G2C $\leq_p$ 2SAT.

Proof:

- **Intuitive idea:**
  - Each node in G2C corresponds to a variable in 2SAT.

Polynomial Time Reductions (continued)

Theorem 2: G2C $\leq_p$ 2SAT.

Proof:

- **Intuitive idea:**
  - Each node in G2C corresponds to a variable in 2SAT.
  - **Blue** corresponds to 1 and **Red** corresponds to 0.

Theorem 2: G2C $\leq_p$ 2SAT.

Proof:

- **Intuitive idea:**
  - Each node in G2C corresponds to a variable in 2SAT.
  - **Blue** corresponds to 1 and **Red** corresponds to 0.
  - Coloring condition enforced by setting up appropriate clauses.

Polynomial Time Reductions (continued)

Theorem 2: G2C $\leq_p$ 2SAT.

Proof:

- **Intuitive idea:**
  - Each node in G2C corresponds to a variable in 2SAT.
  - **Blue** corresponds to 1 and **Red** corresponds to 0.
  - Coloring condition enforced by setting up appropriate clauses.
- **Steps of the Reduction:**

Polynomial Time Reductions (continued)

Theorem 2: G2C $\leq_p$ 2SAT.

Proof:

- **Intuitive idea:**
  - Each node in G2C corresponds to a variable in 2SAT.
  - **Blue** corresponds to 1 and **Red** corresponds to 0.
  - Coloring condition enforced by setting up appropriate clauses.
- **Steps of the Reduction:**
  - G2C Instance I: $G(V, E)$, where $V = \{v_1, v_2, \ldots, v_n\}$.

Theorem 2: G2C $\leq_p$ 2SAT.

Proof:

- **Intuitive idea:**
  - Each node in G2C corresponds to a variable in 2SAT.
  - **Blue** corresponds to 1 and **Red** corresponds to 0.
  - Coloring condition enforced by setting up appropriate clauses.
- **Steps of the Reduction:**
  - G2C Instance I: $G(V, E)$, where $V = \{v_1, v_2, \ldots, v_n\}$.
  - For each node v_i, create a Boolean variable x_i, $1 \leq i \leq n$.

Theorem 2: G2C $\leq_p$ 2SAT.

Proof:

- **Intuitive idea:**
  - Each node in G2C corresponds to a variable in 2SAT.
  - **Blue** corresponds to 1 and **Red** corresponds to 0.
  - Coloring condition enforced by setting up appropriate clauses.
- **Steps of the Reduction:**
  - G2C Instance I: $G(V, E)$, where $V = \{v_1, v_2, \ldots, v_n\}$.
  - For each node v_i, create a Boolean variable x_i, $1 \leq i \leq n$.
  - For each edge $\{v_i, v_j\}$ in G, create two clauses: $(x_i \vee x_j)$ and $(\overline{x_i} \vee \overline{x_j})$.

Polynomial Time Reductions (continued)

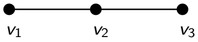
Theorem 2: G2C $\leq_p$ 2SAT.

Proof:

- **Intuitive idea:**
  - Each node in G2C corresponds to a variable in 2SAT.
  - **Blue** corresponds to 1 and **Red** corresponds to 0.
  - Coloring condition enforced by setting up appropriate clauses.
- **Steps of the Reduction:**
  - G2C Instance I: $G(V, E)$, where $V = \{v_1, v_2, \ldots, v_n\}$.
  - For each node v_i, create a Boolean variable x_i, $1 \leq i \leq n$.
  - For each edge $\{v_i, v_j\}$ in G, create two clauses: $(x_i \vee x_j)$ and $(\overline{x_i} \vee \overline{x_j})$.
  - Resulting 2SAT instance I' has $|V|$ variables and $2|E|$ clauses.

Polynomial Time Reductions (continued)

Example:

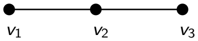


Boolean variables: x_1, x_2, x_3

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

- **Time used for reduction:** $O(|V| + |E|)$

Example:



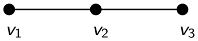
Boolean variables: x_1, x_2, x_3

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

- **Time used for reduction:** $O(|V| + |E|)$
- **Reason for the Choice of Clauses:**

Polynomial Time Reductions (continued)

Example:



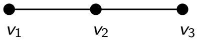
Boolean variables: x_1, x_2, x_3

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

- **Time used for reduction:** $O(|V| + |E|)$
- **Reason for the Choice of Clauses:**
  - For edge $\{v_i, v_j\}$, v_i and v_j must have different colors.

Polynomial Time Reductions (continued)

Example:



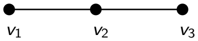
Boolean variables: x_1, x_2, x_3

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

- **Time used for reduction:** $O(|V| + |E|)$
- **Reason for the Choice of Clauses:**
  - For edge $\{v_i, v_j\}$, v_i and v_j must have different colors.
  - So, corresponding Boolean variables x_i and x_j must have *different* values.

Polynomial Time Reductions (continued)

Example:



Boolean variables: x_1, x_2, x_3

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

- **Time used for reduction:** $O(|V| + |E|)$
- **Reason for the Choice of Clauses:**
  - For edge $\{v_i, v_j\}$, v_i and v_j must have different colors.
  - So, corresponding Boolean variables x_i and x_j must have *different* values.
  - In every assignment that satisfies both $(x_i \vee x_j)$ and $(\overline{x_i} \vee \overline{x_j})$, x_i and x_j have *different* values.

Proof of Theorem 2 (continued):

- **Part 1:** Suppose 2SAT instance I' has a solution.

Proof of Theorem 2 (continued):

- **Part 1:** Suppose 2SAT instance I' has a solution.
- **Goal:** Construct a solution to the G2C instance I (i.e., find a 2-coloring of G).

Proof of Theorem 2 (continued):

- **Part 1:** Suppose 2SAT instance I' has a solution.
- **Goal:** Construct a solution to the G2C instance I (i.e., find a 2-coloring of G).
    - For $1 \leq i \leq n$: if variable x_i is set to 1, assign color **Blue** to node v_i; otherwise, assign color **Red** to v_i.

Proof of Theorem 2 (continued):

- **Part 1:** Suppose 2SAT instance I' has a solution.
- **Goal:** Construct a solution to the G2C instance I (i.e., find a 2-coloring of G).
  - For $1 \leq i \leq n$: if variable x_i is set to 1, assign color **Blue** to node v_i; otherwise, assign color **Red** to v_i.
  - Why is this a valid 2-coloring?

Proof of Theorem 2 (continued):

- **Part 1:** Suppose 2SAT instance I' has a solution.
- **Goal:** Construct a solution to the G2C instance I (i.e., find a 2-coloring of G).
    - For $1 \leq i \leq n$: if variable x_i is set to 1, assign color **Blue** to node v_i; otherwise, assign color **Red** to v_i.
    - Why is this a valid 2-coloring?
        - Consider any edge $\{v_i, v_j\}$.

Proof of Theorem 2 (continued):

- **Part 1:** Suppose 2SAT instance I' has a solution.
- **Goal:** Construct a solution to the G2C instance I (i.e., find a 2-coloring of G).
  - For $1 \leq i \leq n$: if variable x_i is set to 1, assign color **Blue** to node v_i; otherwise, assign color **Red** to v_i.
  - Why is this a valid 2-coloring?
    - Consider any edge $\{v_i, v_j\}$.
    - The two chosen clauses ensure that variables x_i and x_j have *different* values.

Proof of Theorem 2 (continued):

- **Part 1:** Suppose 2SAT instance I' has a solution.
- **Goal:** Construct a solution to the G2C instance I (i.e., find a 2-coloring of G).
  - For $1 \leq i \leq n$: if variable x_i is set to 1, assign color **Blue** to node v_i; otherwise, assign color **Red** to v_i.
  - Why is this a valid 2-coloring?
    - Consider any edge $\{v_i, v_j\}$.
    - The two chosen clauses ensure that variables x_i and x_j have *different* values.
    - So, v_i and v_j have different colors.

Proof of Theorem 2 (continued):

Part 2: Suppose G2C instance I has a solution.

Goal: Construct a solution to the 2SAT instance I'.

- Proof similar to that of Part 1. (**Exercise**)

Why are reductions useful?

Lemma 2: Suppose $P \leq_p Q$.

- If Q is efficiently solvable, then so is P.

Polynomial Time Reductions (continued)

Proof of Theorem 2 (continued):

Part 2: Suppose G2C instance I has a solution.

Goal: Construct a solution to the 2SAT instance I'.

- Proof similar to that of Part 1. (**Exercise**)

Why are reductions useful?

Lemma 2: Suppose $P \leq_p Q$.

- If Q is efficiently solvable, then so is P.
- If P is not efficiently solvable, then so is Q.

Polynomial Time Reductions (continued)

Notes:

- The reduction from G2C to 2SAT shows that G2C is efficiently solvable (since 2SAT has an efficient algorithm).
- The reduction from MVC to MIS can be used to conclude that MIS is also **NP**-complete (since MVC is **NP**-complete).
- Thus, reductions are useful to obtain efficient algorithms as well as to prove hardness results.

Polynomial Time Reductions (continued)

What we know about NP-complete problems:

- **NP**-complete problems are the "hardest" ones in **NP**.
- The problems are all "equivalent": they are all efficiently solvable or none of them is efficiently solvable.
- We don't know which of these possibilities is true.
- General conjecture: **NP**-complete problems are not efficiently solvable. (All known algorithms for **NP**-complete problems have exponential running times.)
- **NP**-complete problems are said be "computationally intractable".

Steps to Prove **NP**-completeness

Goal: To prove that Problem Q is **NP**-complete.

- Show that Q is in **NP**. (This step shows the **membership** in **NP**.)
- Identify a suitable problem P which is known to be **NP**-complete.
- Show that $P \leq_p Q$. This shows the **NP-hardness** of Q.

Example:

- It is easy to show that the MIS problem is in **NP**.
- We showed that MVC $\leq_p$ MIS (Theorem 1).
- Since MVC is **NP**-complete, so is MIS.

Important Advice for Reductions

- To show that a problem P is "easy" (i.e., efficiently solvable):

Important Advice for Reductions

- To show that a problem P is "easy" (i.e., efficiently solvable):
  1. You must first identify a problem Q that is known to be easy.

Important Advice for Reductions

- To show that a problem P is "easy" (i.e., efficiently solvable):
  1. You must first identify a problem Q that is known to be easy.
  2. Show that $P \leq_p Q$.

Important Advice for Reductions

- To show that a problem P is "easy" (i.e., efficiently solvable):
  1. You must first identify a problem Q that is known to be easy.
  2. Show that $P \leq_p Q$.

- To show that a problem P is "hard" (i.e., **NP**-complete):

Important Advice for Reductions

- To show that a problem P is "easy" (i.e., efficiently solvable):
  1. You must first identify a problem Q that is known to be easy.
  2. Show that $P \leq_p Q$.

- To show that a problem P is "hard" (i.e., **NP**-complete):
  1. You must first identify with a problem Q that is known to be **NP**-complete.

Important Advice for Reductions

- To show that a problem P is "easy" (i.e., efficiently solvable):
  1. You must first identify a problem Q that is known to be easy.
  2. Show that $P \leq_p Q$.

- To show that a problem P is "hard" (i.e., **NP**-complete):
  1. You must first identify with a problem Q that is known to be **NP**-complete.
  2. Show that $Q \leq_p P$.