15.1 NP-completeness

Recap

Definition 1. NP is the family of decision problems whose YES instances have a polynomial-sized certificate.

Definition 2. A polynomial-time reduction of a problem \( \pi \) to a problem \( \pi' \) is a polynomial-time computable function \( f : \text{instances}(\pi) \to \text{instances}(\pi') \) such that \( S \) is a YES instance of problem \( \pi \) iff \( f(S) \) is a YES instance of problem \( \pi' \).

Example:
\( \pi \): Min Cardinality Vertex Cover
Given: A graph \( G = (V, E), k \in \mathbb{Z}_+ \)
Goal: Does there exist a vertex cover in \( G \) of size \( \leq k \)?
Recall: \( U \subseteq V \) is a vertex cover in \( G \) if every edge in \( G \) has at least one vertex in \( U \).

\( \pi' \): Max Cardinality Stable set
Given: A graph \( G = (V, E), r \in \mathbb{Z}_+ \)
Goal: Does there exist a stable set in \( G \) of size \( \geq r \)?
Recall: A set \( S \subseteq V \) is a stable set in \( G \) if no edge of \( G \) has both its end vertices in \( S \).

Proposition 3. There exists a polynomial-time reduction from \( \pi \) to \( \pi' \) and vice-versa.

Proof. \( \pi \rightarrow \pi' \):

Given an instance of the vertex cover problem \( (G, k) \), consider the mapping \( f(G, r) = (G, |V(G)| - k) \). Then

\[
G \text{ has a vertex cover } U \text{ of size at most } k.
\]
\[
\iff G - U \text{ has no edges.}
\]
\[
\iff V(G) - U \text{ is a stable set in } G.
\]
\[
\iff G \text{ has a stable set of size at least } |V(G)| - k.
\]

The mapping \( f \) is polynomial-time computable.

\( \pi' \rightarrow \pi \):
Given an instance of stable set \((G, r)\), consider the mapping \(f(G, k) = (G, |V(G)| - r)\). Then,

\[
G \text{ has a stable set } S \text{ of size at least } r.
\]

\[
\iff \text{All edges of } G \text{ are between vertices in } V(G) - S \text{ and } V(G).
\]

\[
\iff V(G) - S \text{ is a vertex cover in } G \text{ of size at most } |V(G)| - r.
\]

\[
\iff G \text{ has a vertex cover } U \text{ of size at most } |V(G)| - r.
\]

The mapping \(f\) is polynomial-time computable.

The family of problems in NP that are at least as difficult as all problems in NP (in terms of computation time) are known as NP-complete problems.

**Definition 4.** A problem \(\pi \in \text{NP}\) is **NP-complete** if every problem \(\pi' \in \text{NP}\) has a polynomial-time reduction to \(\pi\).

![Figure 15.1: Problem \(\pi\) is NP-complete if \(\pi\) is in NP and all problems in NP reduce to problem \(\pi\).](image)

If there exists a polynomial-time algorithm for an NP-complete problem, then we have a polynomial-time algorithm for all problems in NP. So it is sufficient to design polynomial-time algorithms for NP-complete problems.

Suppose \(\pi\) is NP-complete. If (1) \(\pi' \in \text{NP}\) and (2) there exist a polynomial-time reduction from \(\pi\) to \(\pi'\), then \(\pi\) is also NP-complete.

![Figure 15.2: If an NP-complete problem \(\pi\) reduces to a problem \(\pi' \in \text{NP}\), Then \(\pi'\) is also NP-complete.](image)

We saw that the theory of NP-completeness is for decision problems. How about optimization problems?
Definition 5. An optimization problem whose decision version is NP-complete is called \textit{NP-hard}.

We will next see that NP-complete problems indeed exist. In fact, the satisfiability problem is NP-complete.

Satisfiability (SAT)

Given: \( n \) boolean variables \( z_1, \ldots, z_n \), \( m \) clauses \( C_1, \ldots, C_m \), where each \( C_i = L_{i_1} \lor L_{i_2} \lor \cdots \lor L_{i_j} \) where \( L_{i_k} \in \{ z_{i_k}, \bar{z}_{i_k} \} \). Here, \( \lor \) corresponds to boolean OR.

Goal: Does there exist an assignment of \( \{0,1\} \) values to the variables \( z_1, \ldots, z_n \) such that all clauses evaluate to 1?

Example: Consider a SAT instance with variables: \( z_1, z_2, z_3 \) and clauses:

\[
C_1 = z_1 \lor z_2 \lor \bar{z}_3 \\
C_2 = \bar{z}_1 \lor z_2 \\
C_3 = \bar{z}_1 \lor \bar{z}_2 \lor z_3.
\]

For this instance, \( z_1 = 1, z_2 = 1, z_3 = 1 \) is a satisfying assignment.

Note that SAT is in NP. A remarkable result by Cook shows that SAT is NP-complete.

Theorem 6 (Cook). SAT is NP-complete.

This tells us that SAT is at least as difficult as all problems in NP. How about integer programs?

Recall: Decision-BIP

<table>
<thead>
<tr>
<th>Given: ( A, b )</th>
<th>Goal: Does there exist ( x \in {0,1}^n ) such that ( Ax \leq b )?</th>
</tr>
</thead>
</table>

Theorem 7. Decision-BIP is NP-complete.

Proof. (i) Decision-BIP is in NP:

Suppose an instance \((A, b)\) has a feasible \( \bar{x} \). Then \( \bar{x} \) is a polynomial-sized certificate because \( \text{size}(\bar{x}) = n \).

(ii) Decision-BIP is NP-complete:

We need to show that there exists an NP-complete problem \( \pi \) such that \( \pi \) is polynomial time reducible to decision-BIP. We will show that SAT reduces to Decision-BIP.

Let \((\text{vars}: z_1, \ldots, z_n, \text{clauses}: C_1, \ldots, C_m)\) be an instance of SAT. Consider the mapping \( f(z_1, \ldots, z_n, C_1, \ldots, C_m) \) that gives a Decision-BIP instance

\[
\sum_{i \in [n]: z_i \in C_j} x_i + \sum_{i \in [n]: \bar{z}_i \in C_j} (1 - x_i) \geq 1 \quad \forall j \in [m]
\]

\[
x_i \in \{0,1\} \quad \forall i \in [n].
\]

Let \( A \) be the constraint matrix, \( b \) be the RHS of the above inequalities. Then, \( A \in \{0,1,-1\}^{m \times n}, b \in \{1,-1,0,\ldots,-(n-1)\}^m \). Therefore, \( \text{size}(A, b) = \text{poly(size(SAT instance)))} \) and \( z \) is
a satisfying assignment iff $x = z$ satisfies the constraints. Therefore, the SAT instance $(z_1, \ldots, z_n, C_1, \ldots, C_m)$ is a YES instance iff the Decision-BIP instance $f(z_1, \ldots, z_n, C_1, \ldots, C_m)$ is a YES instance. Therefore, $f$ is a polynomial-time reduction from SAT to Decision-BIP.

\[ \square \]

**Corollary 7.1.** *BIP is NP-hard.*

Note that Decision-BIP can be reduced to Decision-IP and Decision-MIP. Moreover, Decision-IP and Decision-MIP are also in NP (we will not see a proof of this non-trivial fact). Consequently, MIP and IP are also NP-hard problems.

We have thus seen that *Integer Programming Problem*, the main focus of this course, is at least as difficult as all problems in NP.

### 15.2 Optimization and Separation

We were alluding to connections between efficient optimization and *efficient separation*. Now that we understand what it means to be efficient, let us make this connection between optimization and separation a bit more precise.

Recall that we saw linear programs which had exponential number of constraints, e.g, max weight forest problem, which is a special case of the matroid optimization problem.

**Recall: Max weight forest problem**

\[
\begin{align*}
\text{max} \quad & \sum_{e \in E} w_e x_e \\
\text{s.t.} \quad & \sum_{e \in F} x_e \leq r(F) \quad F \subseteq E \\
& x_e \geq 0 \quad \forall e \in E
\end{align*}
\]  

(15.1)

We saw that extreme point optimal solutions to this LP will be integral and correspond to the indicator vectors of a max weight forest. But how do we solve this LP? How can we solve LPs in which the number constraints is very large (relative to the number of variables)? In the above example, the number of constraints is exponential in the number of variables, i.e., exponential in the size of input which is a graph with weights on the edges.

**Intuition:** We may not need to write down all constraints of the LP to be able to solve it efficiently. See Figure 15.2.
Figure 15.3: Some of the constraints are irrelevant along the objective direction of interest.

To translate this intuition towards a polynomial time algorithm for solving LPs with large number of constraints, we first need an input framework for describing LPs with large number of constraints (without explicitly writing them down). The separation problem gives such a framework.

15.2.1 Separation Problem (over an implicitly given polyhedra)

Given: A bounded rational polyhedron \( P \subseteq \mathbb{R}^n \), a rational vector \( \bar{x} \in \mathbb{R}^n \)

Goal: Verify if \( \bar{x} \in P \) and if not, then find a rational vector \( c \in \mathbb{R}^n \) and a rational value \( \delta \in \mathbb{R} \) such that

\[
    c^T x < \delta \quad \forall x \in P \quad \text{and} \quad c^T \bar{x} > \delta.
\]

Implicitly given polyhedron: e.g., the polyhedron described by the constraints of (15.1) can be given implicitly by just specifying \( G \).

15.2.2 Optimization Problem (over an implicitly given polyhedron)

Given: A bounded rational polyhedron \( P \subseteq \mathbb{R}^n \) and a rational vector \( w \in \mathbb{R}^n \)

Goal: Verify if \( P \) is empty and if not, find \( x^* \in P \) maximizing \( w^T x \).

A landmark result in optimization shows that

**Theorem 8** (Grotschel-Lovasz-Schrijver). [informal version] The separation problem is polynomial-time solvable iff the optimization problem is polynomial-time solvable.

GLS’ result is a rather powerful result that allows us to solve LPs with very large number of constraints. An LP with very large number of constraints may be specified using an efficient oracle which will solve the separation problem for the polyhedron associated with the constraints of the LP. Given this oracle, GLS’ result tells us that we can solve the LP! So, we do not need to know all constraints of the LP explicitly.

**Caution:** GLS’s result is a proof of concept. It gives an efficient algorithm for the optimization problem if the separation problem is solvable but this is not the fastest algorithm. It should be seen as an encouraging sign in the search for efficient algorithms. Upon encountering an algorithm for the optimization problem through a separation algorithm, one should look for direct optimization algorithms, i.e., study the problem further to find fast algorithms.
15.3 Linear Equations Integer Feasibility Problem (LEIF)

Given a constraint matrix $A \in \mathbb{R}^n$, and a RHS vector $b \in \mathbb{R}^m$, we have the following problems:

- Linear Equations Feasibility: Does there exist $x \in \mathbb{R}^n : Ax = b$? (Gaussian Elimination)
- Linear Program: Does there exist $x \in \mathbb{R}^n : Ax \leq b$? (Ellipsoid Algorithm)
- Integer Program: Does there exist $x \in \mathbb{Z}^n : Ax \leq b$? (NP-complete)
- LEIF: Does there exist $x \in \mathbb{Z}^n : Ax = b$?

We know how to solve the linear equations feasibility problem, namely, by Gaussian Elimination. How about the linear equations integer feasibility problem?

**Plot twist:** Unlike Integer Programming which is NP-hard, we will next see that LEIF is solvable efficiently.

Let us see an example to understand how we would go about solving this.

**Example:** Consider the system:

$$
\begin{bmatrix}
2 & 1 & 6 & 4 \\
7 & 2 & 5 & 5 \\
8 & 3 & 33 & 10
\end{bmatrix}
\begin{bmatrix}
x_1 \\
x_2 \\
x_3 \\
x_4
\end{bmatrix}
=
\begin{bmatrix}
1 \\
3 \\
1
\end{bmatrix}
$$

(15.2)

Does there exist $x_1, x_2, x_3, x_4 \in \mathbb{Z}$ satisfying System (15.2)?

We can conclude that it has no integral solution as follows.

\[ \exists x \in \mathbb{Z}^4 \text{ satisfying System } 15.2 \]

\[ \iff \exists x \in \mathbb{Z}^4 : \begin{bmatrix}
1 & 6 & 2 & 4 \\
2 & 5 & 7 & 5 \\
3 & 33 & 8 & 10
\end{bmatrix}
x = \begin{bmatrix}
1 \\
3 \\
1
\end{bmatrix} \]

\[ \iff \exists x \in \mathbb{Z}^4 : \begin{bmatrix}
1 & 0 & 0 & 0 \\
2 & -7 & 3 & -3 \\
3 & 15 & 2 & -2
\end{bmatrix}
x = \begin{bmatrix}
1 \\
3 \\
1
\end{bmatrix} \quad \begin{cases}
c_2 \leftarrow c_2 - 6c_1 \\
c_3 \leftarrow c_3 - 2c_1 \\
c_4 \leftarrow c_4 - 4c_1
\end{cases}
\]

\[ \iff \exists x \in \mathbb{Z}^4 : \begin{bmatrix}
1 & 0 & 0 & 0 \\
2 & -1 & 3 & 0 \\
3 & 19 & 2 & 0
\end{bmatrix}
x = \begin{bmatrix}
1 \\
3 \\
1
\end{bmatrix} \quad \begin{cases}
c_2 \leftarrow c_2 + 2c_3 \\
c_4 \leftarrow c_4 + c_3
\end{cases}
\]

\[ \iff \exists x \in \mathbb{Z}^4 : \begin{bmatrix}
1 & 0 & 0 & 0 \\
2 & 1 & 0 & 0 \\
3 & -19 & 59 & 0
\end{bmatrix}
x = \begin{bmatrix}
1 \\
3 \\
1
\end{bmatrix} \quad \begin{cases}
c_2 \leftarrow -c_2 \\
c_3 \leftarrow c_3 + 3c_2
\end{cases}
\]

\[ \iff x_1 = 1 \]

\[ 2x_1 + x_2 = 3, \quad \text{i.e., } x_2 = 2 \]

\[ 3x_1 - 19x_2 + 59x_3 = 17 \implies \text{No integral solution} \]

Therefore, there is no integral solution for System (15.2).