Towards More NP-Complete Problems

Starting with \textit{Sat}, one can readily show more problems \( P \) to be NP-complete, each time performing two steps:

1. Show that \( P \in \text{NP} \)
2. Find a known NP-complete problem \( P' \) and reduce \( P' \leq_p P \)

Thousands of problems have now been shown to be NP-complete. (See Garey and Johnson for an early survey)

In this course:

\[
\begin{align*}
\text{Sat} & \leq_p \text{3-Sat} \\ 
\text{3-Sat} & \leq_p \text{Dir. Hamiltonian Path} \\ 
\text{3-Sat} & \leq_p \text{Subset Sum} \\ 
\text{3-Sat} & \leq_p \text{Knapsack}
\end{align*}
\]

NP-Completeness of 3-Sat

3-Sat: Satisfiability of formulae in CNF with \( \leq 3 \) literals per clause

**Theorem 8.1:** 3-Sat is NP-complete.

**Proof:** Hardness by reduction \textit{Sat} \( \leq_p \text{3-Sat} \):

- Given: \( \varphi \) in CNF
- Construct \( \varphi' \) by replacing clauses \( C_i = (L_1 \lor \cdots \lor L_k) \) with \( k > 3 \) by

\[
C'_i := (L_1 \lor Y_1) \land (\neg Y_1 \lor L_2 \lor Y_2) \land \cdots \land (\neg Y_{k-1} \lor L_k)
\]

Here, the \( Y_j \) are fresh variables for each clause.
- **Claim:** \( \varphi \) is satisfiable iff \( \varphi' \) is satisfiable.
Example

Let $\varphi := (X_1 \lor X_2 \lor \neg X_3 \lor X_4) \land (\neg X_4 \lor \neg X_2 \lor X_3 \lor \neg X_1)$

Then $\varphi' := (X_1 \lor Y_1) \land (\neg Y_1 \lor X_2 \lor Y_2) \land (\neg Y_2 \lor \neg X_3 \lor Y_3) \land (\neg Y_3 \lor X_4) \land (\neg X_4 \lor Z_1) \land (\neg Z_1 \lor \neg X_2 \lor Z_2) \land (\neg Z_2 \lor X_5 \lor Z_3) \land (\neg Z_3 \lor \neg X_1)$

Proving NP-Completeness of $\text{3-Sat}$

"$\Rightarrow$" Show that if $\varphi'$ is satisfiable then $\varphi$ is satisfiable.

Suppose $\beta$ is a satisfying assignment for $\varphi'$ then $\beta$ satisfies $\varphi$: 

Let $C := (L_1 \lor \cdots \lor L_k)$ be a clause of $\varphi$ 

(1) If $k \leq 3$ then $C$ is a clause of $\varphi$

(2) If $k > 3$ then $C' = (L_1 \lor Y_1) \land (\neg Y_1 \lor L_2 \lor Y_2) \land \cdots \land (\neg Y_{k-1} \lor L_k)$ in $\varphi'$

$\beta$ must satisfy at least one $L_i$, $1 \leq i \leq k$

Case (2) follows since, if $\beta(L_i) = 0$ for all $i \leq k$ then $C'$ can be reduced to

$C' = (Y_1) \land (\neg Y_1 \lor Y_2) \land \cdots \land (\neg Y_{k-1})$

$Y_1 \land (Y_1 \rightarrow Y_2) \land \cdots \land (Y_{k-2} \rightarrow Y_{k-1}) \land \neg Y_{k-1}$

which is not satisfiable.

NP-Completeness of $\text{Directed Hamiltonian Path}$

Input: A directed graph $G$.

Problem: Is there a directed path in $G$ containing every vertex exactly once?

Theorem 8.2: Directed Hamiltonian Path is NP-complete.

Proof:

(1) Directed Hamiltonian Path $\in$ NP;

Take the path to be the certificate.

(2) Directed Hamiltonian Path is NP-hard:

$3$-Sat $\leq_p$ Directed Hamiltonian Path
Digression: How to design reductions

**Task:** Show that problem $P$ (Directed Hamiltonian Path) is NP-hard.

- Arguably, the most important part is to decide where to start from. That is, which problem to reduce to Directed Hamiltonian Path?
- **Considerations:**
  - Is there an NP-complete problem similar to $P$? (for example, Clique and Independent Set)
  - It is not always beneficial to choose a problem of the same type (for example, reducing a graph problem to a graph problem)
    - For instance, Clique, Independent Set are “local” problems (is there a set of vertices inducing some structure)
    - Hamiltonian Path is a global problem (find a structure – the Hamiltonian path – containing all vertices)

**How to design the reduction:**

- Does your problem come from an optimisation problem? If so: a maximisation problem? a minimisation problem?
- Learn from examples, have good ideas.

NP-Completeness of Directed Hamiltonian Path

**Input:** A directed graph $G$.

**Problem:** Is there a directed path in $G$ containing every vertex exactly once?

**Theorem 8.2:** Directed Hamiltonian Path is NP-complete.

**Proof:**

1. **Directed Hamiltonian Path $\in$ NP:**
   Take the path to be the certificate.

2. **Directed Hamiltonian Path is NP-hard:**
   3-Sat $\leq_p$ Directed Hamiltonian Path

Towards More NP-Complete Problems

Starting with Sat, one can readily show more problems $P$ to be NP-complete, each time performing two steps:

1. Show that $P \in$ NP
2. Find a known NP-complete problem $P'$ and reduce $P' \leq_p P$

Thousands of problem have now been shown to be NP-complete. (See Garey and Johnson for an early survey)

In this course:

$$\leq_p \text{Clique} \leq_p \text{Independent Set}$$

$$\text{Sat} \leq_p \text{3-Sat} \leq_p \text{Dir. Hamiltonian Path}$$

$$\leq_p \text{Subset Sum} \leq_p \text{Knapsack}$$
NP-Completeness of **Subset Sum**

**Subset Sum**

Input: A collection $S = \{a_1, \ldots, a_k\}$ and a target integer $t$.

Problem: Is there a subset $T \subseteq S$ such that $\sum_{i \in T} a_i = t$?

**Theorem 8.4: Subset Sum is NP-complete.**

**Proof:**

1. **Subset Sum $\in$ NP**: Take $T$ to be the certificate.

2. **Subset Sum is NP-hard**: SAT $\leq_p$ Subset Sum

1) This “collection” is supposed to be a multi-set, i.e., we allow the same number to occur several times. The solution “subset” can likewise use numbers multiple times, but not more often than they occurred in the given collection.

SAT $\leq_p$ Subset Sum

**Given:** $\varphi := C_1 \land \cdots \land C_k$ in conjunctive normal form.

(w.l.o.g. at most 9 literals per clause)

Let $X_1, \ldots, X_n$ be the variables in $\varphi$. For each $X_i$, let

$$t_i := a_1 \ldots a_n c_1 \ldots c_k$$

where $a_j := \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases}$ and $c_j := \begin{cases} 1 & X_i \text{ occurs in } C_j \\ 0 & \text{otherwise} \end{cases}$

$$f_i := a_1 \ldots a_n c_1 \ldots c_k$$

where $a_j := \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases}$ and $c_j := \begin{cases} 1 & \neg X_i \text{ occurs in } C_j \\ 0 & \text{otherwise} \end{cases}$

\[ (X_1 \lor X_2 \lor X_3) \land (\neg X_1 \lor \neg X_4) \land (X_4 \lor X_5 \lor \neg X_2 \lor \neg X_3) \]

\[ X_1, X_2, X_3, X_4, X_5 \quad C_1, C_2, C_3 \]

\[ t_f = 1 \begin{array}{cccc} 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \end{array} \]

\[ \begin{array}{cc} m_{1,1} & = 1 \ 0 \ 0 \\ m_{1,2} & = 1 \ 0 \ 0 \\ m_{2,1} & = 0 \ 1 \ 0 \\ m_{2,2} & = 0 \ 0 \ 1 \\ m_{3,1} & = 0 \ 0 \ 1 \\ m_{3,2} & = 0 \ 0 \ 1 \end{array} \]

\[ t = 1 \begin{array}{cccc} 1 & 1 & 1 & 1 & 3 & 2 & 4 \end{array} \]
**NP-Completeness of Subset Sum**

Let $\varphi := \bigwedge C_i$, where $C_i$ are clauses.

**Claim:** If $\varphi$ is satisfiable, then there is $T \subseteq S$ with $\sum s \in T s = t$.

**Proof:**
- **Definition of $S$:** Let $S := \{t_i, f_i \mid 1 \leq i \leq n\} \cup \{m_{i,j} \mid 1 \leq i \leq k, \ 1 \leq j \leq |C_i| - 1\}$
- **Target:** Finally, choose as target $t := a_1 \ldots a_n c_1 \ldots c_k$ where $a_i := 1$ and $c_i := |C_i|

**Example**

Let $C := (X_1 \lor X_2 \lor X_3) \land (\neg X_1 \lor \neg X_4) \land (X_4 \lor X_5 \lor \neg X_2 \lor \neg X_3)$

$X_1 \ X_2 \ X_3 \ X_4 \ X_5 \ C_1 \ C_2 \ C_3

\begin{align*}
f_1 & = 1 0 0 0 0 1 0 0 \\
f_2 & = 1 0 0 0 0 1 0 0 \\
f_3 & = 1 0 0 0 1 0 0 1 \\
f_4 & = 1 0 0 0 1 0 0 1 \\
f_5 & = 1 0 0 0 0 1 0 0 \\
m_{1,1} & = 1 0 0 \\
m_{1,2} & = 1 0 0 \\
m_{3,1} & = 0 1 0 \\
m_{3,2} & = 0 0 1 \\
m_{3,3} & = 0 0 1 \\
\end{align*}

$t = 1 1 1 1 1 3 2 4$

**Show:** If there is $T \subseteq S$ with $\sum s \in T s = t$, then $\varphi$ is satisfiable.

Let $\beta$ be a satisfying assignment for $\varphi$.

**Set**: $T_1 := \{t_i \mid \beta(X_i) = 1, \ 1 \leq i \leq m\}$ \cup $\{f_i \mid \beta(X_i) = 0, \ 1 \leq i \leq m\}$

Further, for each clause $C_i$, let $r_i$ be the number of satisfied literals in $C_i$ (with resp. to $\beta$).

Set $T_2 := \{m_{i,j} \mid 1 \leq i \leq k, \ 1 \leq j \leq |C_i| - r_i\}$

and define $T := T_1 \cup T_2$.

It follows: $\sum s \in T s = t$
Towards More NP-Complete Problems

Starting with \textbf{Sat}, one can readily show more problems \( P \) to be NP-complete, each time performing two steps:

(1) Show that \( P \in \text{NP} \)
(2) Find a known NP-complete problem \( P' \) and reduce \( P \leq_p P' \)

Thousands of problem have now been shown to be NP-complete.

(See Garey and Johnson for an early survey)

In this course:

\[
\begin{array}{c|c}
\text{\( \leq_p \text{Clique} \)} & \text{\( \leq_p \text{Independent Set} \)} \\
\text{\( \text{Sat} \)} & \text{\( \leq_p 3\text{-Sat} \)}
\end{array}
\]

\[
\begin{array}{c|c}
\text{\( \leq_p \text{Subset Sum} \)} & \text{\( \leq_p \text{Knapsack} \)}
\end{array}
\]

\[\text{NP-completeness of } \mathbf{Knapsack}\]

\textbf{Knapsack}

\begin{itemize}
  \item Input: A set \( I := \{1, \ldots, n\} \) of items
  \item each of value \( v_i \) and weight \( w_i \) for \( 1 \leq i \leq n \)
  \item target value \( t \) and weight limit \( \ell \)
\end{itemize}

Problem: Is there a subset \( T \subseteq I \) such that

\[\sum_{i \in T} v_i \geq t \text{ and } \sum_{i \in T} w_i \leq \ell?\]

\textbf{Theorem 8.5: Knapsack is NP-complete.}

\textbf{Proof:}

(1) \textbf{Knapsack} \( \in \text{NP}: \) Take \( T \) to be the certificate.

(2) \textbf{Knapsack} is NP-hard: \textbf{Subset Sum} \( \leq_p \text{Knapsack} \)

\textbf{A Polynomial Time Algorithm for Knapsack}

\textbf{Knapsack} can be solved in time \( O(n\ell) \) using dynamic programming

\textbf{Initialisation:}

- Create an \((\ell + 1) \times (n + 1)\) matrix \( M \)
- Set \( M(w, 0) := 0 \) for all \( 1 \leq w \leq \ell \) and \( M(0, i) := 0 \) for all \( 1 \leq i \leq n \)

\textbf{Computation:}

Assign further \( M(w, i) \) to be the largest total value obtainable by selecting from the first \( i \) items with weight limit \( w \):

For \( i = 0, 1, \ldots, n - 1 \) set \( M(w, i + 1) \) as

\[M(w, i + 1) := \max \{ M(w, i), \ M(w - w_{i+1}, i) + v_{i+1} \}\]

Here, if \( w - w_{i+1} < 0 \) we always take \( M(w, i) \).

\textbf{Acceptance:}

If \( M \) contains an entry \( \geq t \), accept. Otherwise reject.
### Example

Input \( I = \{1, 2, 3, 4\} \) with

<table>
<thead>
<tr>
<th>Values</th>
<th>( v_1 = 1 )</th>
<th>( v_2 = 3 )</th>
<th>( v_3 = 4 )</th>
<th>( v_4 = 2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Weight</td>
<td>( w_1 = 1 )</td>
<td>( w_2 = 1 )</td>
<td>( w_3 = 3 )</td>
<td>( w_4 = 2 )</td>
</tr>
</tbody>
</table>

Weight limit: \( \ell = 5 \) Target value: \( t = 7 \)

<table>
<thead>
<tr>
<th>weight limit ( w )</th>
<th>( i = 0 )</th>
<th>( i = 1 )</th>
<th>( i = 2 )</th>
<th>( i = 3 )</th>
<th>( i = 4 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>4</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>4</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>1</td>
<td>4</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>1</td>
<td>4</td>
<td>8</td>
<td></td>
</tr>
</tbody>
</table>

Set \( M(w, 0) := 0 \) for all \( 1 \leq w \leq \ell \) and \( M(0, i) := 0 \) for all \( 1 \leq i \leq n \) For \( i = 0, 1, \ldots, n-1 \) set \( M(w, i + 1) := \max \{M(w, i), M(w - w_{i+1}, i) + v_{i+1}\} \)

### A Polynomial Time Algorithm for Knapsack

**Knapsack** can be solved in time \( O(n\ell) \) using dynamic programming

1. Create an \((\ell + 1) \times (n + 1)\) matrix \( M \)
2. Set \( M(w, 0) := 0 \) for all \( 1 \leq w \leq \ell \) and \( M(0, i) := 0 \) for all \( 1 \leq i \leq n \)

Computation: Assign further \( M(w, i) \) to be the largest total value obtainable by selecting from the first \( i \) items with weight limit \( w \):

For \( i = 0, 1, \ldots, n-1 \) set \( M(w, i + 1) \) as

\[
M(w, i + 1) := \max \{M(w, i), M(w - w_{i+1}, i) + v_{i+1}\}
\]

Here, if \( w - w_{i+1} < 0 \) we always take \( M(w, i) \).

Acceptance: If \( M \) contains an entry \( \geq t \), accept. Otherwise reject.

### Did we prove \( P = NP? \)

**Summary:**

- **Theorem 5:** Knapsack is NP-complete
- **Knapsack** can be solved in time \( O(n\ell) \) using dynamic programming

**What went wrong?**

**Knapsack**

- **Input:** A set \( I := \{1, \ldots, n\} \) of items each of value \( v_i \) and weight \( w_i \) for \( 1 \leq i \leq n \), target value \( t \) and weight limit \( \ell \)

- **Problem:** Is there \( T \subseteq I \) such that \( \sum_{i \in T} v_i \geq t \) and \( \sum_{i \in T} w_i \leq \ell \)?
Pseudo-Polynomial Time

The previous algorithm is not sufficient to show that Knapsack is in P

• The algorithm fills a \((\ell + 1) \times (n + 1)\) matrix \(M\)
• The size of the input to Knapsack is \(O(n \log \ell)\)

\(\leadsto\) the size of \(M\) is not bounded by a polynomial in the length of the input!

**Definition 8.6 (Pseudo-Polynomial Time):** Problems decidable in time polynomial in the sum of the input length and the value of numbers occurring in the input.

Equivalently: Problems decidable in polynomial time when using unary encoding for all numbers in the input.

• If Knapsack is restricted to instances with \(\ell \leq p(n)\) for a polynomial \(p\), then we obtain a problem in P.
• Knapsack is in polynomial time for unary encoding of numbers.

Beyond NP

The Class coNP

Recall that coNP is the complement class of NP.

**Definition 8.7:**
• For a language \(L \subseteq \Sigma^*\) let \(\overline{L} := \Sigma^* \setminus L\) be its complement
• For a complexity class \(C\), we define \(\text{co}C := \{L \mid L \in C\}\)
• In particular \(\text{coNP} = \{L \mid L \in \text{NP}\}\)

A problem belongs to coNP, if no-instances have short certificates.

**Examples:**
• No Hamiltonian Path: Does the graph \(G\) not have a Hamiltonian path?
• Tautology: Is the propositional logic formula \(\varphi\) a tautology (true under all assignments)?
coNP-completeness

**Definition 8.8:** A language $C \in \text{coNP}$ is coNP-complete, if $L \leq_p C$ for all $L \in \text{coNP}$.

**Theorem 8.9:**
1. $P = \text{coP}$
2. Hence, $P \subseteq \text{NP} \cap \text{coNP}$

Open questions:
- $\text{NP} = \text{coNP}$?
  Most people do not think so.
- $P = \text{NP} \cap \text{coNP}$?
  Again, most people do not think so.

Example: Chess Problems

- **Mate in 3 moves; White's turn**
- **Mate in 262 moves; White's turn**

Summary and Outlook

3-Sat and Hamiltonian Path are also NP-complete

So are SubSet Sum and Knapsack, but only if numbers are encoded efficiently (pseudo-polynomial time)

There do not seem to be polynomial certificates for coNP instances; and for some problems there seem to be certificates neither for instances nor for non-instances

**What's next?**
- Space
- Games
- Relating complexity classes