Towards More NP-Complete Problems

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

(1) Show that $\mathcal{P} \in \text{NP}$

(2) Find a known NP-complete problem $\mathcal{P}'$ and reduce $\mathcal{P}' \leq_p \mathcal{P}$

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

In this course:

\begin{align*}
\text{\leq}_p \text{ CLIQUE} & \quad \text{\leq}_p \text{ INDEPENDENT SET} \\
\text{\textsf{Sat}} \leq_p \text{ 3-Sat} & \quad \leq_p \text{ DIR. HAMILTONIAN PATH} \\
\leq_p \text{ Subset Sum} & \quad \leq_p \text{ Knapsack}
\end{align*}
NP-Completeness of Directed Hamiltonian Path

**Directed Hamiltonian Path**

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

**Theorem 9.1**

**Directed Hamiltonian Path is NP-complete.**

**Proof.**

- **Directed Hamiltonian Path $\in$ NP:** Take the path to be the certificate.
- **Directed Hamiltonian Path is NP-hard:** $3$-Sat $\leq_P$ Directed Hamiltonian Path

Digression: How to design reductions

**Task:** Show that problem $P$ (Dir. 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.

**Proof idea:** (see blackboard for details)

Let $\varphi := \bigwedge_{i=1}^{k} C_i$ and $C_i := (L_{i,1} \lor L_{i,2} \lor L_{i,3})$

- For each variable $X$ occurring in $\varphi$, we construct a directed graph (“gadget”) that allows only two Hamiltonian paths: “true” and “false”
- Gadgets for each variable are “chained” in a directed fashion, so that all variables must be assigned one value
- Clauses are represented by vertices that are connected to the gadgets in such a way that they can only be visited on a Hamiltonian path that corresponds to an assignment where they are true

Details are also given in [Sipser, Theorem 7.46].

**Example 9.2** (see blackboard)

$\varphi := C_1 \land C_2$ where $C_1 := (X \lor \neg Y \lor Z)$ and $C_2 := (\neg X \lor Y \lor \neg Z)$
Towards More NP-Complete Problems

Starting with \( \text{Sat} \), 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:

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

Example

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

\[
\begin{array}{ccccccc}
X_1 & X_2 & X_3 & X_4 & X_5 & C_1 & C_2 & C_3 \\
\hline
f_1 & = & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
f_2 & = & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
f_3 & = & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
f_4 & = & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
f_5 & = & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
f_6 & = & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
m_{h,1} & = & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
m_{h,2} & = & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
m_{h,3} & = & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
m_{h,4} & = & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
m_{h,5} & = & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
n & = & 1 & 1 & 1 & 1 & 1 & 3 & 2 & 4
\end{array}
\]

\[\text{Given: } \varphi := C_1 \land \cdots \land C_k \text{ in conjunctive normal form.} \]

(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

\[
t_i := a_1 \cdots a_n c_1 \cdots c_k \text{ where } a_j := \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases} \text{ and } c_j := \begin{cases} 1 & X_i \text{ occurs in } C_j \\ 0 & \text{otherwise} \end{cases}
\]

\[
f_i := a_1 \cdots a_n c_1 \cdots c_k \text{ where } a_j := \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases} \text{ and } c_j := \begin{cases} 1 & \neg X_i \text{ occurs in } C_j \\ 0 & \text{otherwise} \end{cases}
\]
Example

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

\[
\begin{align*}
X_1 & \quad X_2 & \quad X_3 & \quad X_4 & \quad X_5 & \quad C_1 & \quad C_2 & \quad C_3 \\
1 & \quad 0 & \quad 0 & \quad 0 & \quad 1 & \quad 0 & \quad 0 \\
f_1 & = & 1 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 1 & \quad 0 & \quad 0 \\
f_2 & = & 1 & \quad 0 & \quad 0 & \quad 0 & \quad 1 & \quad 0 & \quad 0 \\
f_3 & = & 1 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 1 & \quad 1 & \quad 0 \\
f_4 & = & 1 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 1 & \quad 0 & \quad 0 \\
f_5 & = & 1 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 1 & \quad 0 & \quad 0 \\
f_6 & = & 1 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 1 & \quad 0 \\
m_{1,1} & = & 1 & \quad 0 & \quad 0 \\
m_{1,2} & = & 1 & \quad 0 & \quad 0 \\
m_{2,1} & = & 0 & \quad 1 & \quad 0 \\
m_{2,2} & = & 0 & \quad 0 & \quad 1 \\
m_{3,1} & = & 0 & \quad 0 & \quad 1 \\
m_{3,2} & = & 0 & \quad 0 & \quad 1 \\
m_{3,3} & = & 0 & \quad 0 & \quad 1 \\
t & = & 1 & \quad 1 & \quad 1 & \quad 1 & \quad 1 & \quad 3 & \quad 2 & \quad 4
\end{align*}
\]

**Claim:** There is \( m \) where \( m \subseteq (\alpha_1 \lor \alpha_2 \lor \cdots \lor \alpha_k) \)

Further for each clause \( C \) take \( r := |C| - 1 \) integers \( m_i, \ldots, m_r \)

where \( m_{i,j} := c_i \ldots c_k \) with \( c_{i} := \begin{cases} 1 & i = \ell \\ 0 & i \neq \ell \end{cases} \)

**Definition of \( S \):** Let

\[
S := \{ t_i, f_i \mid 1 \leq i \leq n \} \cup \{ m_{i,j} \mid 1 \leq i \leq k, \quad 1 \leq j \leq |C| - 1 \}
\]

**Target:** Finally, choose as target

\[
t := a_1 \ldots a_n c_1 \ldots c_k \text{ where } a_i := 1 \text{ and } c_i := |C_i|
\]

**Claim:** There is \( T \subseteq S \) with \( \sum_{\alpha_i \in T} \alpha_i = t \) iff \( \psi \) is satisfiable.

---

**NP-Completeness of SubsetSum**

Let \( \psi := \land C_i \quad C_i: \text{ clauses} \)

Show: If \( \psi \) is satisfiable, then there is \( T \subseteq S \) with \( \sum_{s \in T} s = t \).

Let \( \beta \) be a satisfying assignment for \( \psi \)

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

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

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

and define \( T := T_1 \cup T_2 \).

It follows: \( \sum_{s \in T} s = t \)
NP-Completeness of Subset Sum

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

Let $T \subseteq S$ such that $\sum_{s \in T} s = t$

Define $\beta(X_i) = \begin{cases} 1 & \text{if } t_i \in T \\ 0 & \text{if } f_i \in T \end{cases}$

This is well defined as for all $i$: $t_i \in T$ or $f_i \in T$ but not both.

Further, for each clause, there must be one literal set to 1 as for all $i$, the $m_{ij} \in S$ do not sum up to the number of literals in the clause. $\square$

Knapsack and Strong NP-Completeness

NP-completeness of Knapsack

**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 $\sum_{i \in T} w_i \leq \ell$?

Theorem 9.4

Knapsack is NP-complete.

Proof.

$\triangleright$ Knapsack $\in$ NP: Take $T$ to be the certificate.

$\triangleright$ Knapsack is NP-hard: Subset Sum $\leq_p$ Knapsack
**Subset Sum ≤ₚ Knapsack**

**Subset Sum:**
- **Given:** $S := \{a_1, \ldots, a_n\}$ collection of positive integers $t$ target integer
- **Problem:** Is there a subset $T \subseteq S$ such that $\sum_{a_i \in T} a_i = t$?

**Reduction:** From this input to Subset Sum construct
- set of items $I := \{1, \ldots, n\}$
- weights and values $v_i = w_i = a_i$ for all $1 \leq i \leq n$
- target value $t' := t$ and weight limit $\ell := t$

**Clearly:** For every $T \subseteq S$

$$\sum_{a_i \in T} a_i = t \quad \text{iff} \quad \sum_{a_i \in T} v_i \geq t' = t \quad \sum_{a_i \in T} w_i \leq \ell = t$$

**Hence:** The reduction is correct and in polynomial time.

**Example**

**Input** $I = \{1, 2, 3, 4\}$ with
- Values: $v_1 = 1 \quad v_2 = 3 \quad v_3 = 4 \quad v_4 = 2$
- Weight: $w_1 = 1 \quad w_2 = 1 \quad w_3 = 3 \quad w_4 = 2$

**Weight limit:** $\ell = 5 \quad$ **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>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>4</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>1</td>
<td>4</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>1</td>
<td>4</td>
<td>8</td>
<td>8</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**

**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 $1 \leq i \leq n$

**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.
### 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>max. total value from first $i$ items</th>
</tr>
</thead>
<tbody>
<tr>
<td>$0$</td>
<td>$0$</td>
</tr>
<tr>
<td>$1$</td>
<td>$0$</td>
</tr>
<tr>
<td>$2$</td>
<td>$0$</td>
</tr>
<tr>
<td>$3$</td>
<td>$0$</td>
</tr>
<tr>
<td>$4$</td>
<td>$0$</td>
</tr>
<tr>
<td>$5$</td>
<td>$0$</td>
</tr>
<tr>
<td>$i = 0$</td>
<td>$0$</td>
</tr>
<tr>
<td>$i = 1$</td>
<td>$1$</td>
</tr>
<tr>
<td>$i = 2$</td>
<td>$3$</td>
</tr>
<tr>
<td>$i = 3$</td>
<td>$4$</td>
</tr>
<tr>
<td>$i = 4$</td>
<td>$4$</td>
</tr>
<tr>
<td>$M(w,0) := 0$</td>
<td>$0$</td>
</tr>
</tbody>
</table>

Set $M(w, i) := 0$ 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 $M(w, i+1) := \max \{M(w, i), M(w-w_{i+1}, i) + v_{i+1}\}$

### 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 $O(n \log \ell)$.

$\Rightarrow$ the size of $M$ is not bounded by a polynomial in the length of the input.

**Definition 9.5 (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.

### Strong NP-completeness

**Pseudo Polynomial time:** Algorithms polynomial in the maximum of the input length and the value of numbers occurring in the input.

**Examples:**

- Knapsack
- Subset Sum

**Strong NP-completeness:** Problems which remain NP-complete even if all numbers are bounded by a polynomial in the input length (equivalently: even for unary coding of numbers).

**Examples:**

- Clique
- SAT
- Hamiltonian Cycle
- ...

**Note:** Showing $\text{SAT} \leq_p \text{Subset Sum}$ required exponentially large numbers.
The Class \( \text{coNP} \)

Recall that \( \text{coNP} \) is the complement class of \( \text{NP} \).

**Definition 9.6**

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

A problem belongs to \( \text{coNP} \), 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)?
- ...  

\( \text{coNP} \)-completeness

**Definition 9.7**

A language \( C \in \text{coNP} \) is \( \text{coNP} \)-complete, if \( L \leq_{p} C \) for all \( L \in \text{coNP} \).

**Theorem 9.8**

- \( P = \text{coP} \)
- **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.