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 \( P \in \text{NP} \)
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:

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

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 \( \text{SAT} \leq_p \text{3-Sat} \):

- 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 \( Y_i \) are fresh variables for each clause.

- Claim: \( \varphi \) is satisfiable iff \( \varphi' \) is satisfiable.
Proving NP-Completeness of 3-Sat

"⇐" Show that if ϕ′ is satisfiable then so is ϕ.
Suppose β is a satisfying assignment for ϕ. Then β′ is a satisfying assignment for ϕ′:

ϕ := (X1 ∨ X2 ∨ ¬X3 ∨ X4) ∧ (¬X4 ∨ X2 ∨ ¬X3 ∨ ¬X1)

Then ϕ′ := (X1 ∨ Y1) ∧
(¬Y1 ∨ X2 ∨ Y2) ∧
(¬Y2 ∨ ¬X3 ∨ Y3) ∧
(¬Y3 ∨ X4) ∧
(¬X4 ∨ Z1) ∧
(¬Z1 ∨ ¬X2 ∨ Z2) ∧
(¬Z2 ∨ X5 ∨ Z3) ∧
(¬Z3 ∨ ¬X1)

Proving NP-Completeness of 3-Sat

"⇒" Given ϕ := ∨mi=1 Ci with clauses Ci, show that if ϕ is satisfiable then ϕ′ is satisfiable.
For a satisfying assignment β for ϕ, define an assignment β′ for ϕ′:
For each C := (L1 ∨ ⋮ ∨ Lk), with k > 3, in ϕ there is
C′ = (L1 ∨ Y1) ∧ (¬Y1 ∨ L2 ∨ Y2) ∧ ... ∧ (¬Yk−1 ∨ Lk) in ϕ′

As β satisfies ϕ, there is i ≤ k s.th. β(Li) = 1 i.e.
β(X) = 1 if Li = X
β(X) = 0 if Li = ¬X

β′(Yj) = 1 for j < i
β′(Yj) = 0 for j ≥ i
β′(X) = β(X) for all variables in ϕ

This is a satisfying assignment for ϕ′.
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

Proof (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 8.3: $\varphi := C_1 \land C_2$ where $C_1 := (X \lor \neg Y \lor Z)$ and $C_2 := (\neg X \lor Y \lor \neg Z)$ (see blackboard)

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)
NP-Completeness of **Subset Sum**

**Subset Sum**

Input: A collection of positive integers $S = \{a_1, \ldots, a_k\}$ 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** ∈ NP: Take $T$ to be the certificate.
2. **Subset Sum** is NP-hard: $\text{SAT} \leq_p \text{Subset Sum}$

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.

---

**NP-Completeness of Subset Sum**

**Example**

$$(X_1 \lor X_2 \lor X_3) \land (\neg X_1 \lor \neg X_4) \land (X_4 \lor X_5 \lor \neg X_2 \lor \neg X_3)$$

$$(X_1, X_2, X_3, X_4, X_5, C_1, C_2, C_3)$$

$t_1 = 1 \ 0 \ 0 \ 0 \ 0 \ 1 \ 0 \ 0$

$f_1 = 1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 1 \ 0$

$t_2 = 1 \ 0 \ 0 \ 0 \ 0 \ 1 \ 0 \ 0$

$f_2 = 1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 1 \ 0$

$t_3 = 1 \ 0 \ 0 \ 0 \ 0 \ 1 \ 0 \ 0$

$f_3 = 1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 1 \ 0$

$t_4 = 1 \ 0 \ 0 \ 0 \ 0 \ 1 \ 0 \ 0$

$f_4 = 1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 1 \ 0$

$t_5 = 1 \ 0 \ 0 \ 0 \ 0 \ 1 \ 0 \ 0$

$f_5 = 1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 1 \ 0$

$m_{3,1} = 0 \ 0 \ 1$

$m_{3,2} = 0 \ 0 \ 1$

$m_{3,3} = 0 \ 0 \ 1$

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

---

**SAT ≤p Subset Sum**

**Example**

$$(X_1 \lor X_2 \lor X_3) \land (\neg X_1 \lor \neg X_4) \land (X_4 \lor X_5 \lor \neg X_2 \lor \neg X_3)$$

$$(X_1, X_2, X_3, X_4, X_5, C_1, C_2, C_3)$$

$t_1 = 1 \ 0 \ 0 \ 0 \ 0 \ 1 \ 0 \ 0$

$f_1 = 1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 1 \ 0$

$t_2 = 1 \ 0 \ 0 \ 0 \ 0 \ 1 \ 0 \ 0$

$f_2 = 1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 1 \ 0$

$t_3 = 1 \ 0 \ 0 \ 0 \ 0 \ 1 \ 0 \ 0$

$f_3 = 1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 1 \ 0$

$t_4 = 1 \ 0 \ 0 \ 0 \ 0 \ 1 \ 0 \ 0$

$f_4 = 1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 1 \ 0$

$t_5 = 1 \ 0 \ 0 \ 0 \ 0 \ 1 \ 0 \ 0$

$f_5 = 1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 1 \ 0$

$m_{1,1} = 1 \ 0 \ 0$

$m_{1,2} = 1 \ 0 \ 0$

$m_{2,1} = 0 \ 1 \ 0$

$m_{3,1} = 0 \ 0 \ 1$

$m_{3,2} = 0 \ 0 \ 1$

$m_{3,3} = 0 \ 0 \ 1$

$t = 1 \ 1 \ 1 \ 1 \ 1 \ 3 \ 2 \ 4$
**NP-Completeness of Subset Sum**

**Let** $\varphi := \bigwedge C_i$; clauses

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

**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: $\sum_{s \in T} s = t$

---

Example

$(X_1 \lor X_2 \lor X_3) \land (\neg X_1 \lor \neg X_4) \land (X_4 \lor X_5 \lor \neg X_2 \lor \neg X_3)$

<table>
<thead>
<tr>
<th>$X_1$</th>
<th>$X_2$</th>
<th>$X_3$</th>
<th>$X_4$</th>
<th>$X_5$</th>
<th>(\varphi)</th>
<th>$t$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>t_1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>t_2</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>t_3</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>t_4</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>t_5</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td>m_1, 1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td>m_1, 2</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>m_3, 1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td>m_3, 2</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>m_3, 3</td>
<td>0</td>
</tr>
</tbody>
</table>

$t = 1 1 1 1 1 3 2 4$
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** ∈ **NP**
2. Find a known NP-complete problem **P’** and reduce **P’** ≤_p** P**

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

In this course:

\[ \leq_p \text{ CLIQUE} \leq_p \text{ INDEPENDENT SET} \]
\[ \text{ Sat} \leq_p \text{ 3-Sat} \leq_p \text{ DIR. HAMILTONIAN PATH} \]
\[ \leq_p \text{ SUBSET SUM} \leq_p \text{ KNAPSACK} \]

**SUBSET SUM ≤_p KNAPSACK**

Given: \( S := \{a_1, \ldots, a_n\} \) collection of positive integers

\( n = |S| \)

**Subset Sum:** \( t \) target integer

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

**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 \( T \subseteq S \)

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

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

**NP-completeness of 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 8.5:** KNAPSACK is NP-complete.

**Proof:**

1. **KNAPSACK** ∈ **NP**: Take \( T \) to be the certificate.
2. **KNAPSACK** is NP-hard: **SUBSET SUM** ≤_p **KNAPSACK**

**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

Values: $v_1 = 1$ $v_2 = 3$ $v_3 = 4$ $v_4 = 2$

Weight: $w_1 = 1$ $w_2 = 1$ $w_3 = 3$ $w_4 = 2$

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>$i = 0$</td>
<td>0</td>
</tr>
<tr>
<td>$i = 1$</td>
<td>0</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>7</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

Values: $v_1 = 1$ $v_2 = 3$ $v_3 = 4$ $v_4 = 2$

Weight: $w_1 = 1$ $w_2 = 1$ $w_3 = 3$ $w_4 = 2$

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>$i = 0$</td>
<td>0</td>
</tr>
<tr>
<td>$i = 1$</td>
<td>0</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>7</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 $M(w, i + 1) := \max \{M(w, i), M(w - w_{i+1}, i) + v_{i+1}\}$

**Summary:**

- Theorem 8.5: 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

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

Did we prove $P = NP$?
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.

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 Sat \(\leq_p\) Subset Sum required exponentially large numbers.

The Class coNP

Recall that coNP is the complement class of NP.

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

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