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:

$\text{Sat} \leq_p \text{Clique}$

$\text{Sat} \leq_p \text{Indep. Set}$

$\text{Sat} \leq_p \text{3-Sat}$

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

$\text{Sat} \leq_p \text{Subset Sum}$

$\text{Sat} \leq_p \text{Knapsack}$

\section*{NP-Completeness of 3-Sat}

\textbf{3-Sat}: Satisfiability of formulae in CNF with $\leq 3$ literals per clause

\textbf{Theorem 8.1}: 3-Sat is NP-complete.

\textbf{Proof}: Hardness by reduction $\text{Sat} \leq_p \text{3-Sat}$:

- Given: $\psi$ in CNF
- Construct $\psi'$ 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: $\psi$ is satisfiable iff $\psi'$ is satisfiable.
Example

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

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

Proving NP-Completeness of 3-Sat

"⇒" Show that if \( \phi' \) is satisfiable then \( \phi \) is satisfiable

Suppose \( \beta \) is a satisfying assignment for \( \phi' \)

\( \therefore \beta \) is a satisfying assignment for \( \phi \)

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

\( C' = (Y_1) \land (\neg Y_1 \lor Y_2) \land ... \land (\neg Y_{k-1}) \land (Y_1 \rightarrow Y_2) \land ... \land (Y_{k-2} \rightarrow Y_{k-1}) \land \neg Y_{k-1} \)

which is not satisfiable.

Proving NP-Completeness of 3-Sat

"⇒" Given \( \phi := \land_{i=1}^{m} C_i \) with clauses \( C_i \), show that if \( \phi \) is satisfiable then \( \phi' \) is satisfiable

For a satisfying assignment \( \beta \) for \( \phi \), define an assignment \( \beta' \) for \( \phi' \):

For each \( C := (L_1 \lor \cdots \lor L_k) \), with \( k > 3 \), in \( \phi \) there is

\( C' = (L_1 \lor Y_1) \land (\neg Y_1 \lor L_2 \lor Y_2) \land \cdots \land (\neg Y_{k-1} \lor L_k) \) in \( \phi' \)

As \( \beta \) satisfies \( \phi \), there is \( i \leq k \) s.th. \( \beta(L_i) = 1 \) i.e.

\( \beta(X) = 1 \) if \( L_i = X \)

\( \beta(X) = 0 \) if \( L_i = \neg X \)

\( \beta'(Y_i) = 1 \) for \( j < i \)

\( \beta'(Y_i) = 0 \) for \( j \geq i \)

Set

\( \beta'(Y_i) = \beta(X) \) for all variables in \( \phi \)

This is a satisfying assignment for \( \phi' \)

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:

\( 3\text{-}\text{Sat} \leq_p \text{Directed Hamiltonian Path} \)
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**

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

**Proof:**
(1) **Directed Hamiltonian Path** ∈ 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:

$$
\begin{align*}
&\leq_p \text{Clique} & \leq_p \text{Independent Set} \\
&\text{Sat} \leq_p \text{3-Sat} & \leq_p \text{Dir. Hamiltonian Path} \\
&\leq_p \text{Subset Sum} & \leq_p \text{Knapsack}
\end{align*}
$$
NP-Completeness of \textbf{Subset Sum}

**Subset Sum**

\textbf{Input:} A collection of positive integers 
\[ S = \{a_1, \ldots, a_k\} \] and a target integer \( t \).

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

\textbf{Theorem 8.4:} \textbf{Subset Sum} is NP-complete.

\textbf{Proof:}

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

(2) \textbf{Subset Sum} is NP-hard: \textbf{Sat} \( \leq_p \) \textbf{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.

\textbf{Sat} \( \leq_p \) \textbf{Subset Sum}

\textbf{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 \( c_j := \begin{cases} 1 & \neg X_i \text{ occurs in } C_j \\ 0 & \text{otherwise} \end{cases} \)

\textbf{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 & \; X_2 & \; X_3 & \; X_4 & \; X_5 & \; C_1 & \; C_2 & \; C_3 \\
t_1 & \quad 1 & \quad 0 & \quad 0 & \quad 0 & \quad 1 & \quad 0 & \quad 0 \\
t_2 & \quad 1 & \quad 0 & \quad 0 & \quad 0 & \quad 1 & \quad 0 & \quad 0 \\
t_3 & \quad 1 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 0 \\
t_4 & \quad 1 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 0 \\
t_5 & \quad 1 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 0 \\
m_{1,1} & \quad 0 & \quad 0 & \quad 1 \\
m_{1,2} & \quad 1 & \quad 0 & \quad 0 \\
m_{3,1} & \quad 0 & \quad 1 & \\
m_{3,2} & \quad 0 & \quad 0 \quad 1 \\
m_{3,3} & \quad 0 & \quad 0 \quad 1 \\
t & \quad 1 & \quad 1 & \quad 1 & \quad 1 & \quad 3 & \quad 2 & \quad 4
\end{align*}
**NP-Completeness of ** Subset Sum ****

Let $\varphi := \land C_i$; 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 $T_1 := \{ t_i | \beta(X_i) = 1, \ 1 \leq i \leq m \} \cup \{ f_i | \beta(X_i) = 0, \ 1 \leq i \leq m \}$

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

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

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

It follows: $\sum_{s \in T} s = t$

**Example**

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

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

	$f_1$ = 1 0 0 0 0 1 0 0
	$f_2$ = 1 0 0 0 1 0 0
	$f_3$ = 1 0 0 0 0 1 1
	$f_4$ = 1 0 0 1 0
	$f_5$ = 1 0 0 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**

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 $m_{i,j} \in S$ do not sum up to the number of literals in the clause. □
Towards More NP-Complete Problems

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

(1) Show that \texttt{P} \in \texttt{NP}

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

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

In this course:

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

\textbf{Subset Sum} \leq_p \texttt{Knapsack}

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

\texttt{Subset Sum}:

\( t \) target integer

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

\textbf{Reduction:} From this input to \texttt{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 \( r' := t \) and weight limit \( \ell := t \)

\textbf{Clearly:} For every \( T \subseteq S \)

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

\[
\sum_{a \in T} w_i \leq \ell = t
\]

\textbf{Hence:} The reduction is correct and in polynomial time.

\textbf{NP-completeness of \texttt{Knapsack}}

\begin{itemize}
  \item \texttt{Knapsack} 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}

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

\begin{thm}
\texttt{Knapsack} is NP-complete.
\end{thm}

\textbf{Proof:}

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

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

\textbf{A Polynomial Time Algorithm for \texttt{Knapsack}}

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

\textbf{Initialisation:}

\begin{itemize}
  \item Create an \((\ell + 1) \times (n + 1)\) matrix \( M \)
  \item Set \( M(w, 0) := 0 \) for all \( 1 \leq w \leq \ell \) and \( M(0, i) := 0 \) for all \( 1 \leq i \leq n \)
\end{itemize}

\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
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>0</td>
</tr>
<tr>
<td>$i = 3$</td>
<td>0</td>
</tr>
<tr>
<td>$i = 4$</td>
<td>0</td>
</tr>
</tbody>
</table>

Set $M(w, 0) := 0$ for all $1 \leq w \leq \ell$ and $M(0, i) := 0$ for all $1 \leq i \leq n$
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

Initially:
- 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.

Did we prove $P = NP$?

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 $\sum_{i \in T} w_i \leq \ell$?
Pseudo-Polynomial Time
The previous algorithm is not sufficient to show that \textsc{Knapsack} is in P

- The algorithm fills a \((\ell + 1) \times (n + 1)\) matrix \(M\)
- The size of the input to \textsc{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 \textsc{Knapsack} is restricted to instances with \(\ell \leq p(n)\) for a polynomial \(p\), then we obtain a problem in P.
- \textsc{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:**
- \textsc{Knapsack}
- \textsc{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:**
- \textsc{Clique}
- \textsc{Sat}
- \textsc{Hamiltonian Cycle}
- \ldots

**Note:** Showing \textsc{Sat} \(\leq_p\) \textsc{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 \(\overline{L} := \Sigma^* \setminus L\) be its complement
- For a complexity class \(C\), we define \(\text{co}C := \{L : L \in C\}\)
- In particular \(\text{coNP} = \{L : L \in \text{NP}\}\)

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

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

Beyond NP
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

Example: Chess Problems
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