Are NP Problems Hard?

The Structure of NP

Idea: polynomial many-one reductions define an order on problems

NP-Hardness and NP-Completeness

Definition 7.1:
(1) A language $H$ is NP-hard, if $L \leq_p H$ for every language $L \in$ NP.
(2) A language $C$ is NP-complete, if $C$ is NP-hard and $C \in$ NP.

NP-Completeness

- NP-complete problems are the hardest problems in NP.
- They constitute the maximal class (wrt. $\leq_p$) of problems within NP.
- They are all equally difficult – an efficient solution to one would solve them all.

Theorem 7.2: If $L$ is NP-hard and $L \leq_p L'$, then $L'$ is NP-hard as well.
Proving NP-Completeness

How to show NP-completeness
To show that L is NP-complete, we must show that every language in NP can be reduced to L in polynomial time.

Alternative approach
Given an NP-complete language C, we can show that another language L is NP-complete just by showing that

- C \leq_p L
- L ∈ NP

However: Is there any NP-complete problem at all?

The First NP-Complete Problems

Is there any NP-complete problem at all?
Of course there is: the word problem for polynomial time NTMs!

**Polytime NTM**
Input: A polynomial p, a p-time bounded NTM M, and an input word w.
Problem: Does M accept w (in time p(|w|))?

**Theorem 7.3**: Polytime NTM is NP-complete.

Proof: See exercise.

Further NP-Complete Problem?

**Polytime NTM** is NP-complete, but not very interesting:

- not most convenient to work with
- not of much interest outside of complexity theory

Are there more natural NP-complete problems?
Yes, thousands of them!

The Cook-Levin Theorem
The Cook-Levin Theorem


Proof:
1. SAT ∈ NP
   Take satisfying assignments as polynomial certificates for the satisfiability of a formula.
2. SAT is hard for NP
   Proof by reduction from the word problem for NTMs.

The Cook-Levin Theorem

Given:
- a polynomial $p$
- a $p$-time bounded 1-tape NTM $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}})$
- a word $w$

Intended reduction
Define a propositional logic formula $\varphi_{M, w}$ such that $\varphi_{M, w}$ is satisfiable if and only if $M$ accepts $w$ in time $p(|w|)$.

Note
On input $w$ of length $n := |w|$, every computation path of $M$ is of length $\leq p(n)$ and uses $\leq p(n)$ tape cells.

Idea
Use logic to describe a run of $M$ on input $w$ by a formula.

Proving the Cook-Levin Theorem

Given:
- a polynomial $p$
- a $p$-time bounded 1-tape NTM $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}})$
- a word $w$

Intended reduction
Define a propositional logic formula $\varphi_{M, w}$ such that $\varphi_{M, w}$ is satisfiable if and only if $M$ accepts $w$ in time $p(|w|)$.

Note
On input $w$ of length $n := |w|$, every computation path of $M$ is of length $\leq p(n)$ and uses $\leq p(n)$ tape cells.

Idea
Use logic to describe a run of $M$ on input $w$ by a formula.

Proving Cook-Levin: Encoding Configurations

Use propositional variables for describing configurations:
- $Q_q$ for each $q \in Q$ means “$M$ is in state $q \in Q$”
- $P_i$ for each $0 < i < p(n)$ means “the head is at position $i$”
- $S_{a,i}$ for each $a \in \Gamma$ and $0 < i < p(n)$ means “tape cell $i$ contains symbol $a$”

Represent configuration $(q, P, a_0 \ldots a_{p(n)})$
by assigning truth values to variables from the set

$\overline{C} := \{Q_q, P_i, S_{a,i} \mid q \in Q, a \in \Gamma, 0 < i < p(n)\}$

using the truth assignment $\beta$ defined as

$$\beta(Q_q) := \begin{cases} 1 & s = q \\ 0 & s \neq q \end{cases}$$

$$\beta(P_i) := \begin{cases} 1 & i = p \\ 0 & i \neq p \end{cases}$$

$$\beta(S_{a,i}) := \begin{cases} 1 & a = a_i \\ 0 & a \neq a_i \end{cases}$$

Proving Cook-Levin: Validating Configurations

We define a formula $\text{Conf}(\overline{C})$ for a set of configuration variables $\overline{C} = \{Q_q, P_i, S_{a,i} \mid q \in Q, a \in \Gamma, 0 < i < p(n)\}$ as follows:

$$\text{Conf}(\overline{C}) := \bigvee_{q \in Q} (Q_q \land \bigwedge_{q' \neq q} \neg Q_{q'})$$

“$TM$ in exactly one state $q \in Q$”

$$\land \bigvee_{p < p(n)} (P_p \land \bigwedge_{p' < p} \neg P_{p'})$$

“head in exactly one position $p \leq p(n)$”

$$\land \bigwedge_{0 < i < p(n)} \bigvee_{b \in \Sigma} (S_{a,i} \land \bigwedge_{b' \neq b} \neg S_{b,i})$$

“exactly one $a \in \Gamma$ in each cell”
Proving Cook-Levin: Validating Configurations

For an assignment $\beta$ defined on variables in $C$, define
\[
\text{conf}(C, \beta) := \left\{ (q, p, w_0 \ldots w_{p(n)} ) \mid \beta(Q_q) = 1, \beta(P_p) = 1, \beta(S_{w_i}) = 1 \text{ for all } 0 \leq i < p(n) \right\}
\]

Note: $\beta$ may be defined on other variables besides those in $C$.

**Lemma 7.5:** If $\beta$ satisfies $\text{Conf}(C)$ then $|\text{conf}(C, \beta)| = 1$.

We can therefore write $\text{conf}(C, \beta) = (q, p, w)$ to simplify notation.

Observations:
- $\text{conf}(C, \beta)$ is a potential configuration of $M$, but it may not be reachable from the start configuration of $M$ on input $w$.
- Conversely, every configuration $(q, p, w_0 \ldots w_{p(n)})$ induces a satisfying assignment $\beta$ or which $\text{conf}(C, \beta) = (q, p, w_1 \ldots w_{p(n)})$.

---

Proving Cook-Levin: Start and End

**Defined so far:**
- $\text{Conf}(C)$; $C$ describes a potential configuration
- $\text{Next}(C, C')$; $\text{conf}(C, \beta) \vdash_M \text{conf}(C', \beta)$

**Start configuration:**
For an input word $w = w_0 \ldots w_{p(n)} \in \Sigma^*$, we define:
\[
\text{Start}_{M,w}(C) := \text{Conf}(C) \land Q_q \land P_p \land \bigwedge_{i=0}^{p(n)-1} S_{w_i} \land \bigwedge_{i=0}^{p(n)-1} S_{w_i}^i
\]

Then an assignment $\beta$ satisfies $\text{Start}_{M,w}(C)$ if and only if $C$ represents the start configuration of $M$ on input $w$.

**Accepting stop configuration:**
\[
\text{Acc-Conf}(C) := \text{Conf}(C) \land Q_{accept}
\]

Then an assignment $\beta$ satisfies $\text{Acc-Conf}(C)$ if and only if $C$ represents an accepting configuration of $M$.

---

Proving Cook-Levin: Adding Time

Since $M$ is $p$-time bounded, each run may contain up to $p(n)$ steps. We need one set of configuration variables for each.

**Propositional variables**
- $Q_q$, for all $q \in Q$, $0 \leq t \leq p(n)$ means “at time $t$, $M$ is in state $q$”
- $P_i$, for all $0 \leq i, t \leq p(n)$ means “at time $t$, the head is at position $i$”
- $S_{a,i,d}$, for all $a \in \Gamma$ and $0 \leq i, t \leq p(n)$ means “at time $t$, tape cell $i$ contains symbol $a$”

**Notation**
\[
C := \{ Q_q, P_i, S_{a,i,d} \mid q \in Q, 0 \leq i \leq p(n), a \in \Gamma \}
\]
Proving Cook-Levin: The Formula

Given:
- a polynomial \( p \)
- a \( p \)-time bounded 1-tape NTM \( M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}) \)
- a word \( w \)

We define the formula \( \varphi_{p, M, w} \) as follows:

\[
\varphi_{p, M, w} := \text{Start}_{M, w}(C_0) \land \bigvee_{0 \leq t \leq p(|w|)} \left( \text{Acc-Conf}(C_t) \land \bigwedge_{0 \leq i < t} \text{Next}(C_i, C_{i+1}) \right)
\]

“\( C_0 \) encodes the start configuration” and for some polynomial time \( t \):

“\( M \) accepts after \( t \) steps” and “\( C_0, ..., C_t \) encode a computation path”

\textbf{Lemma 7.7:} \( \varphi_{p, M, w} \) is satisfiable if and only if \( M \) accepts \( w \) in time \( p(|w|) \).

Note that an accepting or rejecting stop configuration has no successor.

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:

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

**Theorem 7.8:** Clique is NP-complete.

**Clique:** Given $G, k$, does $G$ contain a clique of order $\geq k$?

**Proof:**

1. **Clique $\in$ NP**
   
   Take the vertex set of a clique of order $k$ as a certificate.

2. **Clique is NP-hard**
   
   We show $\mathsf{Sat} \leq_p \mathsf{Clique}$
   
   To every CNF-formula $\varphi$ assign a graph $G_\varphi$ and a number $k_\varphi$ such that
   
   $\varphi$ satisfiable $\iff$ $G_\varphi$ contains clique of order $k_\varphi$

---

**NP-Completeness of **Independent Set**

**Theorem 7.10:** Independent Set is NP-complete.

**Input:** An undirected graph $G$ and a natural number $k$

**Problem:** Does $G$ contain $k$ vertices that share no edges (independent set)?

**Proof:** Hardness by reduction $\mathsf{Clique} \leq_p \mathsf{Independent Set}$:

- Given $G := (V, E)$ construct $\overline{G} := (V, \{[u, v] \mid [u, v] \notin E \text{ and } u \neq v\})$
- A set $X \subseteq V$ induces a clique in $G$ iff $X$ induces an independent set in $\overline{G}$.
- Reduction: $G$ has a clique of order $k$ iff $\overline{G}$ has an independent set of order $k$.

$\square$
Summary and Outlook

NP-complete problems are the hardest in NP

Polynomial runs of NTMs can be described in propositional logic (Cook-Levin)

**Clique** and **Independent Set** are also NP-complete

What's next?
- More examples of problems
- The limits of NP
- Space complexities