COMPLEXITY THEORY

Lecture 7: NP Completeness

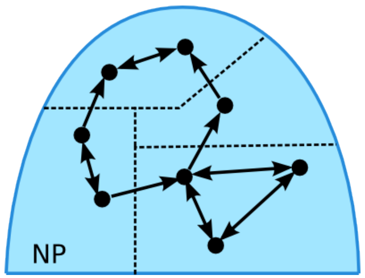
Markus Krötzsch
Knowledge-Based Systems

TU Dresden, 4th Nov 2019

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 $\mathbf{L} \leq_p \mathbf{H}$ for every language $\mathbf{L} \in \mathrm{NP}$.
(2) A language **C** is NP-complete, if **C** is NP-hard and $\mathbf{C} \in \mathrm{NP}$.

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 $\mathbf{L} \leq_p \mathbf{L}'$, then $\mathbf{L}'$ 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** $\in$ 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 $\mathcal{M}$, and an input word w.

Problem: Does $\mathcal{M}$ 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

Theorem 7.4 (Cook 1970, Levin 1973): SAT is NP-complete.

Proof:

(1) SAT $\in$ 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.

$\square$

Proving the Cook-Levin Theorem

Given:

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

Intended reduction

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

Note

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

Idea

Use logic to describe a run of $\mathcal{M}$ 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 "$\mathcal{M}$ is in state $q \in Q$"

P_i for each $0 \leq i < p(n)$ means "the head is at Position i"

$S_{a,i}$ for each $a \in \Gamma$ and $0 \leq i < p(n)$ 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

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

using the truth assignment β defined as

$$\beta(Q_s) := \begin{cases} 1 & s = q \\ 0 & s \neq q \end{cases} \qquad \beta(P_i) := \begin{cases} 1 & i = p \\ 0 & i \neq p \end{cases} \qquad \beta(S_{a,i}) := \begin{cases} 1 & a = a_i \\ 0 & a \neq a_i \end{cases}$$

Proving Cook-Levin: Validating Configurations

We define a formula Conf($\overline{C}$) for a set of configuration variables

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

as follows:

Conf($\overline{C}$) := "the assignment is a valid configuration":

$$\bigvee_{q \in Q}\Big(Q_q \wedge \bigwedge_{q' \neq q} \neg Q_{q'}\Big) \qquad \text{"TM in exactly one state } q \in Q\text{"}$$

$$\wedge \bigvee_{p < p(n)}\Big(P_p \wedge \bigwedge_{p' \neq p} \neg P_{p'}\Big) \qquad \text{"head in exactly one position } p \leq p(n)\text{"}$$

$$\wedge \bigwedge_{0 \leq i < p(n)} \bigvee_{a \in \Gamma}\Big(S_{a,i} \wedge \bigwedge_{b \neq a \in \Gamma} \neg S_{b,i}\Big) \qquad \text{"exactly one } a \in \Gamma \text{ in each cell"}$$

Proving Cook-Levin: Validating Configurations

For an assignment β defined on variables in $\overline{C}$ define

$$\text{conf}(\overline{C},\beta) := \left\{ (q,p,w_0\ldots w_{p(n)}) \;\middle|\; \begin{array}{l} \beta(Q_q)=1, \\ \beta(P_p)=1, \\ \beta(S_{w_i,i})=1 \text{ for all } 0\leq i<p(n) \end{array} \right\}$$

Note: β may be defined on other variables besides those in $\overline{C}$.

Lemma 7.5: If β satisfies Conf($\overline{C}$) then $|\text{conf}(\overline{C},\beta)|=1$.
We can therefore write $\text{conf}(\overline{C},\beta)=(q,p,w)$ to simplify notation.

Observations:

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

Proving Cook-Levin: Transitions Between Configurations

Consider the following formula Next($\overline{C},\overline{C}'$) defined as

$$\text{Conf}(\overline{C})\wedge\text{Conf}(\overline{C}')\wedge\text{NoChange}(\overline{C},\overline{C}')\wedge\text{Change}(\overline{C},\overline{C}').$$

$$\text{NoChange} := \bigvee_{0\leq p<p(n)} \Big(P_p\wedge\bigwedge_{i\neq p, a\in\Gamma}(S_{a,i}\rightarrow S'_{a,i})\Big)$$

$$\text{Change} := \bigvee_{0\leq p<p(n)} \Big(P_p\wedge\bigvee_{\substack{q\in Q\\ a\in\Gamma}}\big(Q_q\wedge S_{a,p}\wedge\bigvee_{(q',b,D)\in\delta(q,a)}(Q'_{q'}\wedge S'_{b,p}\wedge P'_{D(p)})\big)\Big)$$

where $D(p)$ is the position reached by moving in direction D from p.

Lemma 7.6: For any assignment β defined on $\overline{C}\cup\overline{C}'$:

β satisfies Next($\overline{C},\overline{C}'$) if and only if $\text{conf}(\overline{C},\beta)\vdash_{\mathcal{M}}\text{conf}(\overline{C}',\beta)$

Proving Cook-Levin: Start and End

Defined so far:

- Conf($\overline{C}$): $\overline{C}$ describes a potential configuration
- Next($\overline{C},\overline{C}'$): $\text{conf}(\overline{C},\beta)\vdash_{\mathcal{M}}\text{conf}(\overline{C}',\beta)$

Start configuration:

For an input word $w=w_0\cdots w_{n-1}\in\Sigma^*$, we define:

$$\text{Start}_{\mathcal{M},w}(\overline{C}) := \text{Conf}(\overline{C})\wedge Q_{q_0}\wedge P_0\wedge\bigwedge_{i=0}^{n-1}S_{w_i,i}\wedge\bigwedge_{i=n}^{p(n)-1}S_{\llcorner,i}$$

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

Accepting stop configuration:

$$\text{Acc-Conf}(\overline{C}) := \text{Conf}(\overline{C})\wedge Q_{q_{\text{accept}}}$$

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

Proving Cook-Levin: Adding Time

Since $\mathcal{M}$ is p-time bounded, each run may contain up to $p(n)$ steps
$\leadsto$ we need one set of configuration variables for each

Propositional variables

$Q_{q,t}$ for all $q\in Q$, $0\leq t\leq p(n)$ means "at time t, $\mathcal{M}$ is in state $q\in Q$"

$P_{i,t}$ for all $0\leq i,t\leq p(n)$ means "at time t, the head is at position i"

$S_{a,i,t}$ for all $a\in\Gamma$ and $0\leq i,t\leq p(n)$ means "at time t, tape cell i contains symbol a"

Notation

$\overline{C}_t := \{Q_{q,t},\ P_{i,t},\ S_{a,i,t} \mid \quad q\in Q, 0\leq i\leq p(n), \quad a\in\Gamma\}$

Proving Cook-Levin: The Formula

Given:

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

We define the formula $\varphi_{p,\mathcal{M},w}$ as follows:

$$\varphi_{p,\mathcal{M},w} := \text{Start}_{\mathcal{M},w}(\overline{C}_0) \wedge \bigvee_{0\leq t\leq p(n)} \left(\text{Acc-Conf}(\overline{C}_t) \wedge \bigwedge_{0\leq i<t} \text{Next}(\overline{C}_i, \overline{C}_{i+1}) \right)$$

"C_0 encodes the start configuration" and for some polynomial time t:
"$\mathcal{M}$ accepts after t steps" and "$\overline{C}_0, \ldots, \overline{C}_t$ encode a computation path"

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

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

The Cook-Levin Theorem

Theorem 7.4 (Cook 1970, Levin 1973): Sat is NP-complete.

Proof:

(1) **Sat** $\in$ 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.

□

Further NP-complete Problems

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:

	$\leq_p$	**Clique**	$\leq_p$	**Independent Set**
Sat	$\leq_p$	**3-Sat**	$\leq_p$	**Dir. Hamiltonian Path**
	$\leq_p$	**Subset Sum**	$\leq_p$	**Knapsack**

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 **Sat** $\leq_p$ **Clique**

To every CNF-formula φ assign a graph G_φ and a number k_φ such that

$$\varphi \text{ satisfiable} \iff G_\varphi \text{ contains clique of order } k_\varphi$$

Sat $\leq_p$ Clique

To every CNF-formula φ assign a graph G_φ and a number k_φ such that

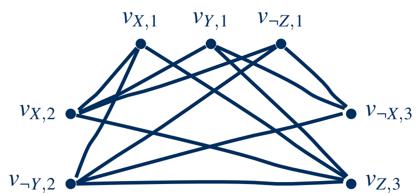
φ satisfiable if and only if G_φ contains clique of order k_φ

Given $\varphi = C_1 \wedge \cdots \wedge C_k$:

- Set $k_\varphi := k$
- For each clause C_j and literal $L \in C_j$ add a vertex $v_{L,j}$
- Add edge $\{v_{L,j}, v_{K,i}\}$ if $i \neq j$ and $L \wedge K$ is satisfiable (that is: if $L \neq \neg K$ and $\neg L \neq K$)

Example 7.9:

$$\underbrace{(X \vee Y \vee \neg Z)}_{C_1} \wedge \underbrace{(X \vee \neg Y)}_{C_2} \wedge \underbrace{(\neg X \vee Z)}_{C_3}$$



Sat $\leq_p$ Clique

To every CNF-formula φ assign a graph G_φ and a number k_φ such that

φ satisfiable if and only if G_φ contains clique of order k_φ

Given $\varphi = C_1 \wedge \cdots \wedge C_k$:

- Set $k_\varphi := k$
- For each clause C_j and literal $L \in C_j$ add a vertex $v_{L,j}$
- Add edge $\{u_{L,j}, v_{K,i}\}$ if $i \neq j$ and $L \wedge K$ is satisfiable (that is: if $L \neq \neg K$ and $\neg L \neq K$)

Correctness:

G_φ has clique of order k iff φ is satisfiable.

Complexity:

The reduction is clearly computable in polynomial time.

NP-Completeness of Independent Set

Independent Set

Input: An undirected graph G and a natural number k

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

Theorem 7.10: Independent Set is NP-complete.

Proof: Hardness by reduction **Clique** $\leq_p$ **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.

□

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