Complexity Theory

NP Completeness

Daniel Borchmann, Markus Krötzsch

Computational Logic

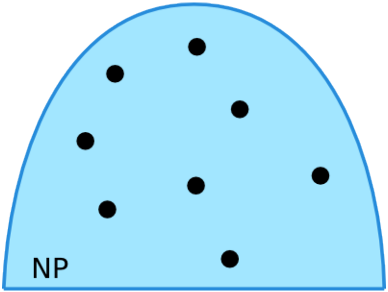
2015-11-17

Review

Are NP Problems Hard?

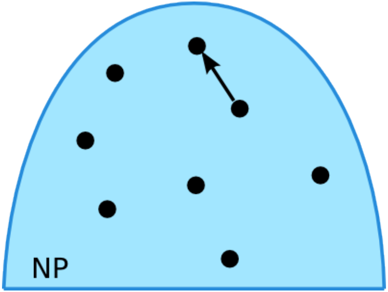
The Structure of NP

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



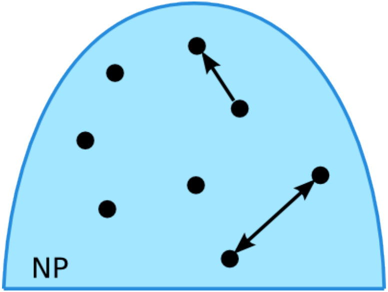
The Structure of NP

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



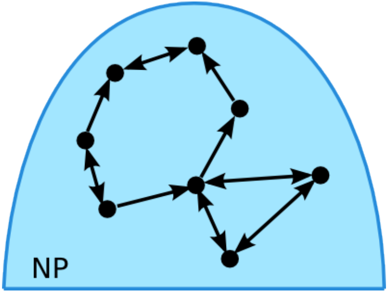
The Structure of NP

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



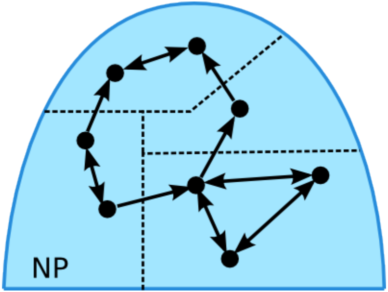
The Structure of NP

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



The Structure of NP

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



NP-Hardness and NP-Completeness

Definition 8.1

- A language $\mathcal{H}$ is NP-hard, if $\mathcal{L} \leq_p \mathcal{H}$ for every language $\mathcal{L} \in \mathrm{NP}$.
- A language $\mathcal{C}$ is NP-complete, if $\mathcal{C}$ is NP-hard and $\mathcal{C} \in \mathrm{NP}$.

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 8.2

If $\mathcal{L}$ is NP-hard and $\mathcal{L} \leq_p \mathcal{L}'$, then $\mathcal{L}'$ is NP-hard as well.

Deterministic vs. Nondeterminsitic Time

Theorem 8.3

$\mathrm{P} \subseteq \mathrm{NP}$, *and also* $\mathrm{P} \subseteq \mathrm{coNP}$.

(Clear since DTMs are a special case of NTMs)

It is not known to date if the converse is true or not.

- Put differently: "If it is easy to check a candidate solution to a problem, is it also easy to find one?"
- Exaggerated: "Can creativity be automated?" (Wigderson, 2006)
- Unresolved since over 35 years of effort
- One of the major problems in computer science and math of our time
- 1,000,000 USD prize for resolving it ("Millenium Problem")
  (might not be much money at the time it is actually solved)

Status of P vs. NP

Many people believe that $\mathrm{P} \neq \mathrm{NP}$

- Main argument: “If $\mathrm{NP} = \mathrm{P}$, someone ought to have found some polynomial algorithm for an NP-complete problem by now.”
- “This is, in my opinion, a very weak argument. The space of algorithms is very large and we are only at the beginning of its exploration.” (Moshe Vardi, 2002)
- Another source of intuition: Humans find it hard to solve NP-problems, and hard to imagine how to make them simpler – possibly “human chauvinistic bravado” (Zeilenberger, 2006)
- There are better arguments, but none more than an intuition

Status of P vs. NP

Many outcomes conceivable:

Status of P vs. NP

Many outcomes conceivable:

- $\mathrm{P} = \mathrm{NP}$ could be shown with a non-constructive proof

Status of P vs. NP

Many outcomes conceivable:

- $\mathrm{P} = \mathrm{NP}$ could be shown with a non-constructive proof
- The question might be independent of standard mathematics (ZFC)

Status of P vs. NP

Many outcomes conceivable:

- $\mathrm{P} = \mathrm{NP}$ could be shown with a non-constructive proof
- The question might be independent of standard mathematics (ZFC)
- Even if $\mathrm{NP} \neq \mathrm{P}$, it is unclear if NP problems require exponential time in a strict sense – many super-polynomial functions exist . . .

Status of P vs. NP

Many outcomes conceivable:

- $\mathrm{P} = \mathrm{NP}$ could be shown with a non-constructive proof
- The question might be independent of standard mathematics (ZFC)
- Even if $\mathrm{NP} \neq \mathrm{P}$, it is unclear if NP problems require exponential time in a strict sense – many super-polynomial functions exist ...
- The problem might never be solved

Status of P vs. NP

Current status in research:

- Results of a poll among 152 experts [Gasarch 2012]:
  - $\mathrm{P} \neq \mathrm{NP}$: 126 (83%)
  - $\mathrm{P} = \mathrm{NP}$: 12 (9%)
  - Don't know or don't care: 7 (4%)
  - Independent: 5 (3%)
  - And 1 person (0.6%) answered: "I don't *want* it to be equal."
- Experts have guessed wrongly in other major questions before
- Over 100 "proofs" show $\mathrm{P} = \mathrm{NP}$ to be true/false/both/neither: `https://www.win.tue.nl/~gwoegi/P-versus-NP.htm`

Proving NP-Completeness

Proving NP-Completeness

How to show NP-completeness

To show that $\mathcal{L}$ is NP-complete, we must show that every language in NP can be reduced to $\mathcal{L}$ in polynomial time.

Alternative approach

Given an NP-complete language C, we can show that another language $\mathcal{L}$ is NP-complete just by showing that

- $C \leq_p \mathcal{L}$
- $\mathcal{L} \in \mathrm{NP}$

Proving NP-Completeness

How to show NP-completeness

To show that $\mathcal{L}$ is NP-complete, we must show that every language in NP can be reduced to $\mathcal{L}$ in polynomial time.

Alternative approach

Given an NP-complete language C, we can show that another language $\mathcal{L}$ is NP-complete just by showing that

- $C \leq_p \mathcal{L}$
- $\mathcal{L} \in \mathrm{NP}$

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

The First NP-Complete Problem

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 $p(|w|)$)?

The First NP-Complete Problem

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 $p(\lvert w \rvert)$)?

Theorem 8.4

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 8.5 (Cook 1970, Levin 1973)

SAT *is* NP*-complete.*

The Cook-Levin Theorem

Theorem 8.5 (Cook 1970, Levin 1973)

SAT *is* NP*-complete.*

Proof.

- SAT $\in \mathrm{NP}$

  Take satisfying assignments as polynomial certificates for the satisfiability of a formula.

The Cook-Levin Theorem

Theorem 8.5 (Cook 1970, Levin 1973)

SAT *is* NP*-complete.*

Proof.

- SAT $\in \mathrm{NP}$

  Take satisfying assignments as polynomial certificates for the satisfiability of a formula.

- SAT is hard for NP

  Proof by reduction from the word problem for NTMs.

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

$S_{a,i}$ for each $a \in \Gamma$ and $0 \leq i \leq p(n)$ 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 $\textsc{Conf}(\overline{C})$ 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:

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

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

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

$$\wedge \bigwedge_{1 \leq i \leq p(n)} \bigvee_{a \in \Gamma} \Big(S_{a,i} \wedge \bigwedge_{b \neq a \in \Gamma} \neg S_{b,i} \Big)$$ "exactly one $a \in \Gamma$ in each cell"

Proving Cook-Levin: Validating Configurations

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

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

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

Proving Cook-Levin: Validating Configurations

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

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

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

Lemma 8.6

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

Proving Cook-Levin: Validating Configurations

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

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

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

Lemma 8.6

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

Observations:

- $\mathrm{conf}(\overline{C},\beta)$ 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 $\mathrm{conf}(\overline{C},\beta) = (q,p,w_1 \ldots w_{p(n)})$.

Proving Cook-Levin: Transitions Between Configurations

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

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

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

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

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

Proving Cook-Levin: Transitions Between Configurations

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

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

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

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

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

Lemma 8.7

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

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

Proving Cook-Levin: Start and End

Defined so far:

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

Proving Cook-Levin: Start and End

Defined so far:

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

Start configuration: Let $w = w_0 \cdots w_{n-1} \in \Sigma^*$ be the input word

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

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

Proving Cook-Levin: Start and End

Defined so far:

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

Start configuration: Let $w = w_0 \cdots w_{n-1} \in \Sigma^*$ be the input word

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

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

Accepting stop configuration:

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

Then an assignment β satisfies $\textsc{Acc-Conf}(\overline{C})$ 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 $a \in \Sigma \dot{\cup} \{\square\}$ 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:

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

"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 comp. path"

Lemma 8.8

$\varphi_{p,\mathcal{M},w}$ 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 8.5 (Cook 1970, Levin 1973)

SAT *is* NP*-complete.*

Proof.

- SAT $\in \mathrm{NP}$

  Take satisfying assignments as polynomial certificates for the satisfiability of a formula.

- 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 $\mathcal{P}$ to be NP-complete, each time performing two steps:

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

(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)

Towards More NP-Complete Problems

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

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

(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:

	$\leq_p$ 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 8.9

Clique *is* NP*-complete.*

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

Proof.

- Clique $\in \mathrm{NP}$

  Take the vertex set of a clique of order k as a certificate.

- Clique is NP-hard

  We show Sat $\leq_p$ Clique

  To every CNF-formula φ assign G_φ, 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 G_φ, k_φ such that

$$\varphi \text{ satisfiable if and only if } G_\varphi \text{ contains clique of order } k_\varphi$$

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 $\neg L \neq K$)

Example 8.10

$(X \vee Y \vee \neg Z) \wedge (X \vee \neg Y) \wedge (\neg X \vee Z)$
See blackboard.

SAT $\leq_p$ CLIQUE

To every CNF-formula φ assign G_φ, 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 8.11

Independent Set is NP-complete.

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 8.11

Independent Set *is* NP*-complete.*

Proof.

Hardness by reduction Clique $\leq_p$ Independent Set:

- Given $G := (V, E)$ construct $\overline{G} := \left(V, \left\{\{u, v\} \mid \{u, v\} \notin E \text{ and } u \neq v\right\}\right)$

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 8.11

INDEPENDENT SET is NP-complete.

Proof.

Hardness by reduction CLIQUE $\leq_p$ INDEPENDENT SET:

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

3-Sat, Hamiltonian Path and SubsetSum

Towards More NP-Complete Problems

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

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

(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:

	$\leq_p$	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 3-SAT

3-SAT: Satisfiability of formulae in CNF with ≤ 3 literals per clause

Theorem 8.12

3-SAT is NP*-complete.*

Proof.

Hardness by reduction SAT $\leq_p$ 3-SAT:

- Given: φ in CNF
- Construct φ' by replacing clauses $C_i = (L_1 \vee \cdots \vee L_k)$ with $k > 3$ by

$$C_i' := (L_1 \vee Y_1) \wedge (\neg Y_1 \vee L_2 \vee Y_2) \wedge \ldots \wedge (\neg Y_{k-1} \vee L_k)$$

  Here, the Y_j are fresh variables for each clause.
- Claim: φ is satisfiable iff φ' is satisfiable.

Example

Let $\varphi := (X_1 \vee X_2 \vee \neg X_3 \vee X_4) \quad \wedge \quad (\neg X_4 \vee \neg X_2 \vee X_5 \vee \neg X_1)$

Then $\varphi' :=$

$$
\begin{aligned}
&(X_1 \vee Y_1) \wedge \\
&(\neg Y_1 \vee X_2 \vee Y_2) \wedge \\
&(\neg Y_2 \vee \neg X_3 \vee Y_3) \wedge \\
&(\neg Y_3 \vee X_4) \wedge \\
&(\neg X_4 \vee Z_1) \wedge \\
&(\neg Z_1 \vee \neg X_2 \vee Z_2) \wedge \\
&(\neg Z_2 \vee X_5 \vee Z_3) \wedge \\
&(\neg Z_3 \vee \neg X_1)
\end{aligned}
$$

Proving NP-Completeness of 3-Sat

"$\Rightarrow$" Given $\varphi := \bigwedge_{i=1}^{m} C_i$ with clauses C_i, 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 $C := (L_1 \vee \cdots \vee L_k)$, with $k > 3$, in φ there is

$$C' = (L_1 \vee Y_1) \ \wedge \ (\neg Y_1 \vee L_2 \vee Y_2) \ \wedge \ldots \ \wedge \ (\neg Y_{k-1} \vee L_k) \text{ in } \varphi'$$

As β satisfies φ, there is $i \leq k$ s.th. $\beta(L_i) = 1$ i.e. $\begin{array}{l} \beta(X) = 1 \text{ if } L_i = X \\ \beta(X) = 0 \text{ if } L_i = \neg X \end{array}$

$$\text{Set} \quad \begin{array}{ll} \beta'(Y_j) = 1 & \text{for } j < i \\ \beta'(Y_j) = 0 & \text{for } j \geq i \\ \beta'(X) = \beta(X) & \text{for all variables in } \varphi \end{array}$$

This is a satisfying asignment for φ'

Proving NP-Completeness of 3-Sat

"$\Leftarrow$" Show that if φ' is satisfiable then so is φ

Suppose β is a satisfying assignment for φ' – then β satisfies φ:

Let $C := (L_1 \vee \cdots \vee L_k)$ be a clause of φ

(1) If $k \leq 3$ then C is a clause of φ

(2) If $k > 3$ then

$$C' = (L_1 \vee Y_1) \wedge (\neg Y_1 \vee L_2 \vee Y_2) \wedge \ldots \wedge (\neg Y_{k-1} \vee L_k) \text{ in } \varphi'$$

β must satisfy at least one L_i, $1 \leq i \leq k$

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

$$\begin{aligned} C' &= (Y_1) \wedge (\neg Y_1 \vee Y_2) \wedge \ldots \wedge (\neg Y_{k-1}) \\ &\equiv Y_1 \wedge (Y_1 \to Y_2) \wedge \ldots \wedge (Y_{k-2} \to Y_{k-1}) \wedge \neg Y_{k-1} \end{aligned}$$

which is not satisfiable. $\square$