Computational Complexity

Lecture 2: NP-completeness and the Cook-Levin Theorem

Ronald de Haan
me@ronalddehaan.eu

University of Amsterdam

April 5, 2024
Recap

What we saw last time..

- Decision problems
- The complexity class P
- Nondeterministic Turing machines
- More complexity classes: EXP, NP, coNP
- Polynomial-time reductions
- NP-hardness and NP-completeness
What will we do today?

- Prove that NP-complete problems exist 😊
- The Cook-Levin Theorem
- Concrete reductions between problems
- Search vs. decision problems
Representing Turing machines as (binary) strings

- We can encode Turing machines into binary strings, such that:
  1. each string \( s \in \{0, 1\}^* \) represents some Turing machine \( M \)
  2. each Turing machine \( M \) is represented by infinitely many strings \( s \in \{0, 1\}^* \)
  3. given a TM \( M \), we can efficiently compute a string \( s \) that represents \( M \)

- Idea:
  - Write out the tuple \( (\Gamma, Q, \delta) \), together with starting and halting states, in an appropriate alphabet, and then encode into binary
  - Allow padding (cf. comments in programming languages)
Proposition

There exists a TM $U$ such that for every $x, s \in \{0, 1\}^*$ it holds that $U(x, s) = M_s(x)$, where $M_s$ is the TM represented by the string $s$.

Moreover, if $M_s$ halts on $x$ in time $T$, then $U(x, s)$ halts in time $C \cdot T \log T$, where $C$ depends only on $s$ (and not on $x$).

- $U$ is an efficient universal Turing machine: it can simulate other TMs in an efficient way.
Our first NP-complete problem

Definition

The decision problem TM-SAT is defined as follows:

\[ 
TM\text{-SAT} = \{ (\alpha, x, 1^n, 1^t) \mid \text{there exists } u \in \{0, 1\}^n \text{ such that } M_\alpha \text{ outputs 1 on input } (x, u) \text{ within } t \text{ steps} \} 
\]

Or, described in a different format:

Input: A binary string \( \alpha \), a binary string \( x \), a unary string \( 1^n \), and a unary string \( 1^t \).

Question: Does there exist a binary string \( u \in \{0, 1\}^n \) such that \( M_\alpha \) outputs 1 on input \( (x, u) \) within \( t \) steps?
TM-SAT is NP-complete

Proposition

TM-SAT is NP-complete

Proof (sketch).

Membership in NP: guess \( u \), and verify by simulating \( \mathbb{M}_\alpha \).

NP-hardness:

Take an arbitrary \( L \in \text{NP} \). Then there exists a polynomial \( p \) and a TM \( \mathbb{M} \) such that for all \( x \in \{0, 1\}^* \) there exists some \( u \in \{0, 1\}^{p(|x|)} \) such that \( \mathbb{M}(x, u) = 1 \) iff \( x \in L \).

Let \( q \) be a polynomial bounding the running time of \( \mathbb{M} \).

Take the reduction \( R \) from \( L \) to TM-SAT where:

\[
R(x) = (\text{repr}(\mathbb{M}), x, 1^{p(|x|)}, 1^{q(|x|)+p(|x|)})
\]
Propositional logic formulas $\varphi$ are built from atomic propositions $x_1, x_2, \ldots$ using Boolean operators $\land, \lor, \rightarrow, \neg$.

For example, $\varphi_1 = (x_1 \lor \neg x_2) \land (\neg x_1 \lor x_3)$.

A truth assignment is a function $\alpha : \text{Vars}(\varphi) \rightarrow \{0, 1\}$ that maps the atomic propositions to 1 (true) or 0 (false).

For example, $\alpha_1 = \{x_1 \mapsto 1, x_2 \mapsto 1, x_3 \mapsto 0\}$.

The truth $\varphi[\alpha]$ of a formula $\varphi$ under a truth assignment $\alpha$ is defined inductively, following the standard meaning of the operators.

For example, $\varphi_1[\alpha_1] = 0$. 
Definition

The decision problem Formula-SAT is defined as follows:

\[ \text{Formula-SAT} = \{ \varphi \mid \varphi \text{ is a propositional logic formula and there exists a satisfying truth assignment } \alpha \text{ for } \varphi \} \]

Or, described in a different format:

**Input:** A propositional logic formula \( \varphi \).

**Question:** Is \( \varphi \) satisfiable?
Definition

The decision problem CNF-SAT is defined as follows:

\[ \text{CNF-SAT} = \{ \varphi \mid \varphi \text{ is a propositional logic formula in CNF and there exists a satisfying truth assignment } \alpha \text{ for } \varphi \} \]

Or, described in a different format:

*Input:* A propositional logic formula \( \varphi \) in CNF.

*Question:* Is \( \varphi \) satisfiable?

- Conjunctive Normal Form (CNF): a conjunction of disjunctions of literals.

- For example: \( \varphi_1 = (x_1 \lor \neg x_2) \land (\neg x_1 \lor x_3) \land (\neg x_2 \lor \neg x_3 \lor x_4) \)
The Cook-Levin Theorem

Theorem (Cook 1971, Levin 1969)

CNF-SAT is NP-complete.
Polynomial-time computation in a picture

For a single-tape TM

For each $t, i \in \{1, \ldots, T\}$ and each $\gamma \in \Gamma$:
introduce a proposition $c_{t, i, \gamma}$

For each $t, i \in \{1, \ldots, T\}$:
introduce a proposition $h_{t, i}$

For each $t \in \{1, \ldots, T\}$ and each $q \in Q$:
introduce a proposition $s_{t, q}$
Proof of Cook-Levin Theorem

- Take an arbitrary $L \in \text{NP}$. Then there exist polynomials $p, q : \mathbb{N} \rightarrow \mathbb{N}$ and a TM $M$ running in time $q(n)$ such that for each $x \in \{0,1\}^*$:
  
  $x \in L$ if and only if there exists $u \in \{0,1\}^{p(|x|)}$ such that $M(x, u) = 1$.

- W.l.o.g., assume that $M$ is single-tape and that $q_{\text{acc}}$ and $q_{\text{rej}}$ are ‘sinks’

- Take $T = q(|x| + p(|x|))$. That is, $T \geq$ running time of $M(x, u)$.

- We will construct a formula $\varphi$ (over the variables $c_{t,i,\gamma}$, $h_{t,i}$, $s_{t,q}$) that is satisfiable if and only if $x \in L$.

- $\varphi$ is the conjunction of several clauses (see next slides).
Proof of Cook-Levin Theorem (ct’d)

- Initialize tape contents:
  - \((c_{1,i,x_i})\) for \(1 \leq i \leq |x|\)
  - \((c_{1,i,0} \lor c_{1,i,1})\) for \(|x| < i \leq |x| + p(|x|)\)
  - \((c_{1,i,\Box})\) for \(|x| + p(|x|) < i \leq T\)

- Other initial conditions:
  - \((h_{1,1})\)
  - \((s_{1,q_{\text{start}}})\)
At most one symbol per cell (at each time):

- \((\neg c_{t,i,\gamma} \lor \neg c_{t,i,\gamma'})\) for \(1 \leq i, t \leq T\) and all \(\gamma, \gamma' \in \Gamma\) with \(\gamma \neq \gamma'\)

At most one tape head position at each time:

- \((\neg h_{t,i} \lor \neg h_{t,i'})\) for \(1 \leq i, i', t \leq T\) with \(i \neq i'\)

At most one state at each time:

- \((\neg s_{t,q} \lor \neg s_{t,q'})\) for \(1 \leq t \leq T\) and \(q, q' \in Q\) with \(q \neq q'\)
Correct transitions.

For $1 \leq i, t \leq T - 1$, $\gamma \in \Gamma$, and $q \in Q$:

- $(c_{t,i,\gamma} \land h_{t,i} \land s_{t,q}) \rightarrow (c_{t+1,i,\gamma'} \land h_{t+1,i} \land s_{t+1,q'})$ if $\delta(q, \gamma) = (q', \gamma', S)$

- $(c_{t,i,\gamma} \land h_{t,i} \land s_{t,q}) \rightarrow (c_{t+1,i,\gamma'} \land h_{t+1,i+1} \land s_{t+1,q'})$ if $\delta(q, \gamma) = (q', \gamma', R)$

- $(c_{t,i,\gamma} \land h_{t,i} \land s_{t,q}) \rightarrow (c_{t+1,i,\gamma'} \land h_{t+1,i-1} \land s_{t+1,q'})$ if $\delta(q, \gamma) = (q', \gamma', L)$
No change when the tape head is away:

\[(c_{t,i,\gamma} \land \neg h_{t,i}) \rightarrow c_{t+1,i,\gamma}\] for \(1 \leq t \leq T - 1, 1 \leq i \leq T\) and \(\gamma \in \Gamma\)

The machine must accept:

\[ST, q_{\text{acc}}\]
The formula $\varphi$ is satisfiable if and only if there exists some $u \in \{0, 1\}^{p(|x|)}$ such that $M(x, u) = 1$, and thus if and only if $x \in L$.

The conjuncts of $\varphi$ can be equivalently rewritten as clauses (of size $\leq 4$)

\[ (a \land b \land c) \rightarrow (d \land e \land f) \iff (\lnot a \lor \lnot b \lor \lnot c \lor d) \land (\lnot a \lor \lnot b \lor \lnot c \lor e) \land (\lnot a \lor \lnot b \lor \lnot c \lor f) \]

Computing $\varphi$ takes polynomial time.

- Polynomial number of atomic propositions and clauses
Definition

The decision problem 3SAT is defined as follows:

\[
3\text{SAT} = \{ \varphi \mid \varphi \text{ is a propositional logic formula in 3CNF and there exists a satisfying truth assignment } \alpha \text{ for } \varphi \} \]

Or, described in a different format:

- **Input:** A propositional logic formula \( \varphi \) in 3CNF.
- **Question:** Is \( \varphi \) satisfiable?

- 3CNF: each clause (disjunction) contains at most 3 literals
3SAT is NP-complete

Theorem (Cook 1971, Levin 1969)

3SAT is NP-complete.

- The formula that we constructed is in 4CNF. So 4SAT is NP-complete. We give a polynomial-time reduction from 4SAT to 3SAT.

- We replace each clause $c = (\ell_1 \lor \ell_2 \lor \ell_3 \lor \ell_4)$ of length 4 by:

$$
(\ell_1 \lor \ell_2 \lor z_c) \land (\neg z_c \lor \ell_3 \lor \ell_4),
$$

where $z_c$ is a fresh variable.

- The resulting formula $\varphi'$ is satisfiable if and only if the original formula $\varphi$ is satisfiable.
The web of reductions

Theorem 2.10 (Lemma 2.11)

∀L ∈ NP

Theorem 2.10 (Lemma 2.14)

SAT

INTEGEROPT

Theorem 2.15

3SAT

Ex 2.17

Exactone 3SAT

3COL

Ex 2.21

THEOREMS

TSP

HAMPATH

dHAMPATH

Ex 2.19

QUADEQ

HAMCYCLE

COMBINATORIAL AUCTION

Ex 2.18

Ex 2.18

Ex 2.22

CLIQUE

VERTEXCOVER

MAXCUT

Ex 2.15

Ex 2.16

Ex 2.17

Ex 2.17

Ex 2.17

Ex 2.21
3COL is NP-complete

Theorem (Karp 1972)

3COL is NP-complete.

We will show NP-hardness by reduction from 3SAT.
Gadgets
for each variable $x_i$

for each clause $c_j$
Example

$$\varphi = (\neg x_1 \lor \neg x_2 \lor x_3)$$
Example

\( \varphi = (\neg x_1 \lor \neg x_2 \lor x_3) \), \( \alpha = \{ x_1 \mapsto 1, x_2 \mapsto 1, x_3 \mapsto 1 \} \)
Example

$\varphi = (\neg x_1 \lor \neg x_2 \lor x_3)$, $\alpha = \{x_1 \mapsto 1, x_2 \mapsto 1, x_3 \mapsto 1\}$
Example
\[ \varphi = (\neg x_1 \lor \neg x_2 \lor x_3), \alpha = \{x_1 \mapsto 1, x_2 \mapsto 1, x_3 \mapsto 1\} \]
Example

\( \varphi = (\neg x_1 \lor \neg x_2 \lor x_3) \), \( \alpha = \{ x_1 \mapsto 1, x_2 \mapsto 1, x_3 \mapsto 0 \} \)
Example

$\varphi = (\neg x_1 \lor \neg x_2 \lor x_3)$, $\alpha = \{x_1 \mapsto 1, x_2 \mapsto 1, x_3 \mapsto 0\}$
Does NP-completeness tell us something useful about the search problems on which our decision problems are based?

**Proposition**

Suppose that $P = NP$. Then for every $L \in NP$ and each verifier $M$ for $L$, there exists a polynomial-time Turing machine $B$ that on input $x \in L$ outputs a certificate $u$ for $x$. 
Hamiltonian cycles in grid graphs

For the homework.

- A grid graph $G$..

..and a Hamiltonian cycle in $G$. 
A Slitherlink instance \( I \)...

..and a solution for \( I \).
Recap

- Prove that NP-complete problems exist 😊
- The Cook-Levin Theorem
- Concrete reductions between problems
- Search vs. decision problems
Next time

- Diagonalization arguments
- Time Hierarchy Theorems
- $P \neq \text{EXP}$