The Cook-Levin Theorem

Matvey Soloviev (Cornell University)

CS 4820, Summer 2020

Today

By using our formal definition of nondeterministic Turing machines, we will prove the existence of a universal subproblem which every problem in NP can be reduced to in polynomial time.

This problem will be provably hard under the assumption that any problem in NP is.

Hardness from reductions, made concrete (1)

Before, we talked about how a reduction from P to Q lets us infer hardness of Q from hardness of P conceptually.

Hardness from reductions, made concrete (1)

Before, we talked about how a reduction from P to Q lets us infer hardness of Q from hardness of P conceptually.

Let's make a concrete instance of the statement.

Hardness from reductions, made concrete (1)

Before, we talked about how a reduction from P to Q lets us infer hardness of Q from hardness of P conceptually.

Let's make a concrete instance of the statement.

Proposition

Suppose P is a problem for which there exists no polynomial-time algorithm, and there is a polynomial-time reduction from P to Q.
Then there is no polynomial-time algorithm for Q.

Hardness from reductions, made concrete (2)

Proof. Suppose not, and there is in fact a polynomial-time algorithm for Q that runs in time $O(p(n))$.

Hardness from reductions, made concrete (2)

Proof. Suppose not, and there is in fact a polynomial-time algorithm for Q that runs in time $O(p(n))$.

Plug this algorithm into the reduction. By definition of a polynomial-time reduction, we obtain a correct algorithm that solves size-n instances of P in time $h(n) + g(n) \cdot O(p(f(n)))$.

Hardness from reductions, made concrete (2)

Proof. Suppose not, and there is in fact a polynomial-time algorithm for Q that runs in time $O(p(n))$.

Plug this algorithm into the reduction. By definition of a polynomial-time reduction, we obtain a correct algorithm that solves size-n instances of P in time $h(n) + g(n) \cdot O(p(f(n)))$.

Since all of f, g and h are polynomials, this is again a polynomial (check this!).

Proof. Suppose not, and there is in fact a polynomial-time algorithm for Q that runs in time $O(p(n))$.

Plug this algorithm into the reduction. By definition of a polynomial-time reduction, we obtain a correct algorithm that solves size-n instances of P in time $h(n) + g(n) \cdot O(p(f(n)))$.

Since all of f, g and h are polynomials, this is again a polynomial (check this!).

But then we actually do have a polynomial-time algorithm for P, contradicting the assumption. $\square$

Boolean formulae

A Boolean formula is an expression generated recursively by the grammar

$$\varphi, \psi ::= \varphi \wedge \psi \mid \varphi \vee \psi \mid \neg\varphi \mid x, y, z, \ldots \mid (\varphi),$$

where $x, y, z, \ldots \in V$ are variable names, which represents a function from assignments to truth values $\{T, F\}$. An assignment is itself a function from V to truth values.

For instance, $\varphi = x \wedge (y \vee z) \to (\neg v)$.

Conjunctive normal form

A Boolean formula is said to be in conjunctive normal form (CNF) if it is of the form

$$(\ell_{1,1} \vee \ldots \ell_{1,n_1}) \wedge \ldots \wedge (\ell_{m,1} \vee \ldots \vee \ell_{m,n_m}),$$

where each $\ell_{i,j}$ is a literal: that is, either a variable $v \in V$ or the negation $\neg v$ of one.

For example, $(x \vee \neg z) \wedge (y \vee z \vee x \vee v) \wedge (x)$.

Conjunctive normal form

A Boolean formula is said to be in conjunctive normal form (CNF) if it is of the form

$$(\ell_{1,1} \vee \ldots \ell_{1,n_1}) \wedge \ldots \wedge (\ell_{m,1} \vee \ldots \vee \ell_{m,n_m}),$$

where each $\ell_{i,j}$ is a literal: that is, either a variable $v \in V$ or the negation $\neg v$ of one.

For example, $(x \vee \neg z) \wedge (y \vee z \vee x \vee v) \wedge (x)$.

Each $(\ell_{i,1} \vee \ldots \vee \ell_{i,n_i})$ is called a clause.

The conjunctive normal form Boolean satisfiability problem (CNF-SAT) is:

Given a Boolean formula in CNF, does there exist any assignment to its variables that makes this formula true?

The CNF-SAT problem

The conjunctive normal form Boolean satisfiability problem (CNF-SAT) is:

Given a Boolean formula in CNF, does there exist any assignment to its variables that makes this formula true?

This problem is in NP: We can write out the assignment (as a string of the form $x \mapsto T, y \mapsto F, \ldots$) and use it as a certificate.

The Cook-Levin Theorem

Theorem (Cook-Levin)

The CNF-SAT problem is NP-complete.

Proof of Cook-Levin: High-level overview (1)

We already know that the problem is in NP, so it just remains to establish that it is NP-hard.

Proof of Cook-Levin: High-level overview (1)

We already know that the problem is in NP, so it just remains to establish that it is NP-hard.

That is, we need to show that there exists a polynomial-time reduction from every problem in NP to CNF-SAT.

Proof of Cook-Levin: High-level overview (2)

We will establish this as follows:

Proof of Cook-Levin: High-level overview (2)

We will establish this as follows:

- Let P be an arbitrary problem in NP.

Proof of Cook-Levin: High-level overview (2)

We will establish this as follows:

- Let P be an arbitrary problem in NP.
- Since P is in NP, there must exist a nondeterministic polynomial-time Turing machine M_P which decides P.

Proof of Cook-Levin: High-level overview (2)

We will establish this as follows:

- Let P be an arbitrary problem in NP.
- Since P is in NP, there must exist a nondeterministic polynomial-time Turing machine M_P which decides P.
- Using our knowledge of this machine, we can write the following reduction:

Proof of Cook-Levin: High-level overview (2)

We will establish this as follows:

- Let P be an arbitrary problem in NP.
- Since P is in NP, there must exist a nondeterministic polynomial-time Turing machine M_P which decides P.
- Using our knowledge of this machine, we can write the following reduction:
    1. Read the input α to P.

Proof of Cook-Levin: High-level overview (2)

We will establish this as follows:

- Let P be an arbitrary problem in NP.
- Since P is in NP, there must exist a nondeterministic polynomial-time Turing machine M_P which decides P.
- Using our knowledge of this machine, we can write the following reduction:
  1. Read the input α to P.
  2. In time polynomial in α, write out a specially prepared CNF Boolean formula $\varphi_{M_P}(\alpha)$ which has a satisfying assignment if and only if M_P accepts the string α.

Proof of Cook-Levin: High-level overview (2)

We will establish this as follows:

- Let P be an arbitrary problem in NP.
- Since P is in NP, there must exist a nondeterministic polynomial-time Turing machine M_P which decides P.
- Using our knowledge of this machine, we can write the following reduction:
  1. Read the input α to P.
  2. In time polynomial in α, write out a specially prepared CNF Boolean formula $\varphi_{M_P}(\alpha)$ which has a satisfying assignment if and only if M_P accepts the string α.
  3. Run our algorithm for CNF-SAT on this formula.

Proof of Cook-Levin: High-level overview (2)

We will establish this as follows:

- Let P be an arbitrary problem in NP.
- Since P is in NP, there must exist a nondeterministic polynomial-time Turing machine M_P which decides P.
- Using our knowledge of this machine, we can write the following reduction:
  1. Read the input α to P.
  2. In time polynomial in α, write out a specially prepared CNF Boolean formula $\varphi_{M_P}(\alpha)$ which has a satisfying assignment if and only if M_P accepts the string α.
  3. Run our algorithm for CNF-SAT on this formula.
  4. Accept if the algorithm accepted. Reject if the algorithm rejected.

Proof of Cook-Levin: High-level overview (3)

The meat of the proof clearly is in the second point: Why should such a formula even exist, and why can we write it out in polynomial time?

Proof of Cook-Levin: High-level overview (3)

The meat of the proof clearly is in the second point: Why should such a formula even exist, and why can we write it out in polynomial time?

It is for the sake of this point that we put in all the work of formally defining NTMs.

Some notational points that will help us stay sane.

Proof of Cook-Levin: Notational preliminaries (1)

Some notational points that will help us stay sane.

- For a finite set S and formulae φ_s for every $s \in S$, $\bigwedge_{s \in S} \varphi_s$ denotes the conjunction $\varphi_{s_1} \wedge \ldots \wedge \varphi_{s_n}$.

Proof of Cook-Levin: Notational preliminaries (1)

Some notational points that will help us stay sane.

- For a finite set S and formulae φ_s for every $s \in S$, $\bigwedge_{s \in S} \varphi_s$ denotes the conjunction $\varphi_{s_1} \wedge \ldots \wedge \varphi_{s_n}$.
- Likewise for $\bigvee$.

Proof of Cook-Levin: Notational preliminaries (1)

Some notational points that will help us stay sane.

- For a finite set S and formulae φ_s for every $s \in S$, $\bigwedge_{s \in S} \varphi_s$ denotes the conjunction $\varphi_{s_1} \wedge \ldots \wedge \varphi_{s_n}$.
- Likewise for $\bigvee$.
- By definition, the implication $\varphi \to \psi$ is equivalent to $\neg\varphi \wedge \psi$.

Proof of Cook-Levin: Notational preliminaries (1)

Some notational points that will help us stay sane.

- For a finite set S and formulae φ_s for every $s \in S$, $\bigwedge_{s \in S} \varphi_s$ denotes the conjunction $\varphi_{s_1} \wedge \ldots \wedge \varphi_{s_n}$.
- Likewise for $\bigvee$.
- By definition, the implication $\varphi \to \psi$ is equivalent to $\neg\varphi \wedge \psi$.
- Recall de Morgan's laws: $\neg(\varphi \wedge \psi) \triangleq \neg\varphi \vee \neg\psi$; $\neg(\varphi \vee \psi) \triangleq \neg\varphi \wedge \neg\psi$.

Proof of Cook-Levin: Notational preliminaries (2)

From these, it follows that we can turn a big implication between conjunctions into a conjunction of clauses as follows:

$$\bigwedge_{s \in S} \varphi_s \to \bigwedge_{t \in T} \psi_t$$

Proof of Cook-Levin: Notational preliminaries (2)

From these, it follows that we can turn a big implication between conjunctions into a conjunction of clauses as follows:

$$\begin{aligned} & \bigwedge_{s \in S} \varphi_s \to \bigwedge_{t \in T} \psi_t \\ \triangleq \; & \bigwedge_{t \in T} \left(\bigwedge_{s \in S} \varphi_s \to \psi_t \right) \end{aligned}$$

Proof of Cook-Levin: Notational preliminaries (2)

From these, it follows that we can turn a big implication between conjunctions into a conjunction of clauses as follows:

$$
\begin{aligned}
& \bigwedge_{s \in S} \varphi_s \to \bigwedge_{t \in T} \psi_t \\
\triangleq \quad & \bigwedge_{t \in T} \left(\bigwedge_{s \in S} \varphi_s \to \psi_t \right) \\
\triangleq \quad & \bigwedge_{t \in T} (\neg \varphi_{s_1} \vee \ldots \vee \neg \varphi_{s_n} \vee \psi_t)
\end{aligned}
$$

Proof of Cook-Levin: Notational preliminaries (2)

From these, it follows that we can turn a big implication between conjunctions into a conjunction of clauses as follows:

$$
\begin{aligned}
& \bigwedge_{s \in S} \varphi_s \to \bigwedge_{t \in T} \psi_t \\
\triangleq \quad & \bigwedge_{t \in T} \left(\bigwedge_{s \in S} \varphi_s \to \psi_t \right) \\
\triangleq \quad & \bigwedge_{t \in T} (\neg\varphi_{s_1} \vee \ldots \vee \neg\varphi_{s_n} \vee \psi_t) \\
\triangleq \quad & (\neg\varphi_{s_1} \vee \ldots \vee \neg\varphi_{s_n} \vee \psi_{t_1}) \wedge \ldots \wedge (\ldots \vee \psi_{t_m}).
\end{aligned}
$$

Proof of Cook-Levin: Notational preliminaries (2)

From these, it follows that we can turn a big implication between conjunctions into a conjunction of clauses as follows:

$$
\begin{aligned}
& \bigwedge_{s \in S} \varphi_s \to \bigwedge_{t \in T} \psi_t \\
\triangleq \quad & \bigwedge_{t \in T} \left(\bigwedge_{s \in S} \varphi_s \to \psi_t \right) \\
\triangleq \quad & \bigwedge_{t \in T} (\neg\varphi_{s_1} \vee \ldots \vee \neg\varphi_{s_n} \vee \psi_t) \\
\triangleq \quad & (\neg\varphi_{s_1} \vee \ldots \vee \neg\varphi_{s_n} \vee \psi_{t_1}) \wedge \ldots \wedge (\ldots \vee \psi_{t_m}).
\end{aligned}
$$

Note in particular that the size of the resulting CNF formula is a product of the sizes of S and T.

Proof of Cook-Levin: Mid-level overview (1)

We know that there's a polynomial p such that M_P halts on α in $p(|\alpha|)$ steps.

Proof of Cook-Levin: Mid-level overview (1)

We know that there's a polynomial p such that M_P halts on α in $p(|\alpha|)$ steps.

At any time t, any possible configuration of the NTM M_P consists of at most $p(|\alpha|) + |\alpha|$ non-$\sqcup$ cells on the tape, one state and one integer encoding the position of the head. (We can only fill at most one cell per step!)

Proof of Cook-Levin: Mid-level overview (1)

We know that there's a polynomial p such that M_P halts on α in $p(|\alpha|)$ steps.

At any time t, any possible configuration of the NTM M_P consists of at most $p(|\alpha|) + |\alpha|$ non-$\sqcup$ cells on the tape, one state and one integer encoding the position of the head. (We can only fill at most one cell per step!)

Based on this, we will create Boolean variables for each time $0 \leq t \leq p(\alpha)$ encoding

Proof of Cook-Levin: Mid-level overview (1)

We know that there's a polynomial p such that M_P halts on α in $p(|\alpha|)$ steps.

At any time t, any possible configuration of the NTM M_P consists of at most $p(|\alpha|) + |\alpha|$ non-$\sqcup$ cells on the tape, one state and one integer encoding the position of the head. (We can only fill at most one cell per step!)

Based on this, we will create Boolean variables for each time $0 \leq t \leq p(\alpha)$ encoding

- the contents of the potentially non-empty cells of the tape,

Proof of Cook-Levin: Mid-level overview (1)

We know that there's a polynomial p such that M_P halts on α in $p(|\alpha|)$ steps.

At any time t, any possible configuration of the NTM M_P consists of at most $p(|\alpha|) + |\alpha|$ non-$\sqcup$ cells on the tape, one state and one integer encoding the position of the head. (We can only fill at most one cell per step!)

Based on this, we will create Boolean variables for each time $0 \leq t \leq p(\alpha)$ encoding

- the contents of the potentially non-empty cells of the tape,
- the state q and

Proof of Cook-Levin: Mid-level overview (1)

We know that there's a polynomial p such that M_P halts on α in $p(|\alpha|)$ steps.

At any time t, any possible configuration of the NTM M_P consists of at most $p(|\alpha|) + |\alpha|$ non-$\sqcup$ cells on the tape, one state and one integer encoding the position of the head. (We can only fill at most one cell per step!)

Based on this, we will create Boolean variables for each time $0 \leq t \leq p(\alpha)$ encoding

- the contents of the potentially non-empty cells of the tape,
- the state q and
- the position of the head.

Proof of Cook-Levin: Mid-level overview (1)

We know that there's a polynomial p such that M_P halts on α in $p(|\alpha|)$ steps.

At any time t, any possible configuration of the NTM M_P consists of at most $p(|\alpha|) + |\alpha|$ non-$\sqcup$ cells on the tape, one state and one integer encoding the position of the head. (We can only fill at most one cell per step!)

Based on this, we will create Boolean variables for each time $0 \leq t \leq p(\alpha)$ encoding

- the contents of the potentially non-empty cells of the tape,
- the state q and
- the position of the head.

This should be a polynomial number!

Then, we will generate a polynomial number of clauses that encode the following statements about the variables:

Proof of Cook-Levin: Mid-level overview (2)

Then, we will generate a polynomial number of clauses that encode the following statements about the variables:

- At time 0, the state is the initial state, the tape contains $\triangleright \alpha \sqcup \sqcup \dots$ and the head points at the start.

Proof of Cook-Levin: Mid-level overview (2)

Then, we will generate a polynomial number of clauses that encode the following statements about the variables:

- At time 0, the state is the initial state, the tape contains $\triangleright\alpha \sqcup \sqcup \ldots$ and the head points at the start.
- At each time *t*, we are in a unique configuration: there's a well-defined state, tape contents and head position.

Proof of Cook-Levin: Mid-level overview (2)

Then, we will generate a polynomial number of clauses that encode the following statements about the variables:

- At time 0, the state is the initial state, the tape contains $\triangleright \alpha \sqcup \sqcup \dots$ and the head points at the start.
- At each time t, we are in a unique configuration: there's a well-defined state, tape contents and head position.
- At each time $t > 0$, the configuration must follow from the configuration at time $t-1$ by a possible transition of M_P.

Proof of Cook-Levin: Mid-level overview (2)

Then, we will generate a polynomial number of clauses that encode the following statements about the variables:

- At time 0, the state is the initial state, the tape contains $\triangleright\alpha \sqcup \sqcup \ldots$ and the head points at the start.
- At each time t, we are in a unique configuration: there's a well-defined state, tape contents and head position.
- At each time $t > 0$, the configuration must follow from the configuration at time $t - 1$ by a possible transition of M_P.
- At some point, we are in an accepting state.

Why will this work?

Key trick: While the cone of possible configurations of M_P is of exponential size, a single path of computation is only polynomial. Assignments to our variables only encode a single path of computation, but asking for existence of a satisfying assignment $\Leftrightarrow$ asking for existence of an accepting path $\Leftrightarrow$ NTM acceptance.

Let $m = p(|\alpha|)$. We create variables:

$S_{i,q}$ for all $0 \leq i \leq m$, $q \in Q$ "at time i, state is q"

Proof of Cook-Levin: The details (1)

Let $m = p(|\alpha|)$. We create variables:

$S_{i,q}$	for all $0 \leq i \leq m$, $q \in Q$	"at time i, state is q"
$T_{i,j,\sigma}$	for all $0 \leq i, j \leq m$, $\sigma \in \Sigma$	"at time i, tape[j] is σ"

Proof of Cook-Levin: The details (1)

Let $m = p(|\alpha|)$. We create variables:

$$\begin{array}{rcl} S_{i,q} & \text{for all } 0 \le i \le m,\ q \in Q & \text{“at time } i\text{, state is } q\text{”} \\ T_{i,j,\sigma} & \text{for all } 0 \le i,j \le m,\ \sigma \in \Sigma & \text{“at time } i\text{, tape}[j] \text{ is } \sigma\text{”} \\ H_{i,j} & \text{for all } 0 \le i,j \le m & \text{“at time } i\text{, position is } j\text{”.} \end{array}$$

Proof of Cook-Levin: The details (2)

Now, create clauses.

Initial state at time 0:

Proof of Cook-Levin: The details (2)

Now, create clauses.

Initial state at time 0:

$$S_{0,q_0} \wedge H_{0,0} \wedge T_{0,0,\triangleright} \wedge \bigwedge_{1 \leq n \leq |\alpha|} T_{0,n,\alpha_n} \wedge \bigwedge_{|\alpha| < n \leq m} T_{0,n,\sqcup}.$$

Proof of Cook-Levin: The details (3)

Consistency (well-defined configuration) at every time $i \leq m$:

$$\bigwedge_{i \leq m} \bigwedge_{q \in Q} \left(S_{i,q} \to \bigwedge_{q \neq q' \in Q} \neg S_{i,q'} \right),$$

$$\bigwedge_{i \leq m} \bigwedge_{j \leq m} \left(H_{i,j} \to \bigwedge_{j \neq j' \leq m} \neg H_{i,j'} \right),$$

$$\bigwedge_{i \leq m} \bigwedge_{j \leq m} \bigwedge_{\sigma \in \Sigma} \left(T_{i,j,\sigma} \to \bigwedge_{\sigma \neq \sigma' \in \Sigma} \neg T_{i,j,\sigma'} \right).$$

Transition: at every time $i < m$, we must choose a transition in the set $\delta(q, \sigma)$ and move the head, update the tape and change states accordingly.

$$\bigwedge_{i<m} \bigwedge_{j \le m} \bigwedge_{q \in Q} \bigwedge_{\sigma \in \Sigma} \Bigg(H_{i,j} \wedge S_{i,q} \wedge T_{i,j,\sigma} \\ \to \bigvee_{(q',\sigma',\mathrm{dir}) \in \delta(q,\sigma)} (H_{i+1,j'} \wedge S_{i+1,q'} \wedge T_{i+1,j,\sigma'}) \Bigg),$$

where

$$j' = \begin{cases} j+1 & \text{if dir} = R \\ j & \text{if } j = 1 \text{ and dir} = L \\ j-1 & \text{if dir} = L \end{cases}$$

is the updated position of the head;

Transition consistency: Only the symbol under the head changes!

$$\bigwedge_{i<m} \bigwedge_{j\leq m} \left(H_{i,j} \to \bigwedge_{j\neq j' \leq m} \bigwedge_{\sigma \in \Sigma} \left(T_{i,j',\sigma} \to T_{i+1,j',\sigma} \right) \right)$$

Proof of Cook-Levin: The details (6)

Acceptance:

$$\bigvee_{i \leq m} S_{i,q_{\mathrm{yes}}}.$$

Proof of Cook-Levin: The details (7)

Now just need to establish that M_P accepts α iff the CNF-SAT instance we constructed has a satisfying assignment.

Both directions are easy:

Proof of Cook-Levin: The details (7)

Now just need to establish that M_P accepts α iff the CNF-SAT instance we constructed has a satisfying assignment.

Both directions are easy:

- If M_P accepts α, there must be an accepting path of NTM transitions. Convert to an assignment and show that it's satisfying.

Proof of Cook-Levin: The details (7)

Now just need to establish that M_P accepts α iff the CNF-SAT instance we constructed has a satisfying assignment.

Both directions are easy:

- If M_P accepts α, there must be an accepting path of NTM transitions. Convert to an assignment and show that it's satisfying.
- If there is a satisfying assignment, show that there is a well-defined conversion to a sequence of possible NTM transitions that ends at q_{yes}.

Proof of Cook-Levin: The details (7)

Now just need to establish that M_P accepts α iff the CNF-SAT instance we constructed has a satisfying assignment.

Both directions are easy:

- If M_P accepts α, there must be an accepting path of NTM transitions. Convert to an assignment and show that it's satisfying.
- If there is a satisfying assignment, show that there is a well-defined conversion to a sequence of possible NTM transitions that ends at q_{yes}.

And we're done! □

Corollary

If even one problem in NP can not be solved in polynomial time, then CNF-SAT can not be solved in polynomial time.