Complexity Theory
Nondeterministic Polynomial Time

Daniel Borchmann, Markus Krötzsch

Computational Logic

2015-11-11
The Class \textbf{NP}
Beyond $\text{PTime}$

- We have seen that the class $\text{PTime}$ provides a useful model of “tractable” problems.
- This includes $2\text{-Sat}$ and $2\text{-Colourability}$.
- But what about $3\text{-Sat}$ and $3\text{-Colourability}$?
- No polynomial time algorithms for these problems are known.
- On the other hand . . .
Verifying Solutions

For many seemingly difficult problems, it is easy to verify the correctness of a “solution” if given.

- **Satisfiability** – a satisfying assignment
- **k-Colourability** – a $k$-colouring
- **Sudoku** – a completed puzzle
Verifiers

Definition 7.1

A Turing machine $M$ which halts on all inputs is called a **verifier for a language** $L$ if

$$L = \{w \mid M \text{ accepts } (w\#c) \text{ for some string } c\}$$

The string $c$ is called a **certificate** (or **witness**) for $w$.

$M$ is a **polynomial-time verifier** for $L$ if $M$ is polynomially time bounded and

$$L = \{w \mid M \text{ accepts } (w\#c) \text{ for some string } c \text{ with } |c| \leq p(|w|)\}$$

for some fixed polynomial $p$.

Notation: $\#$ is a new separator symbol not used in words or certificates.
The Class **NP**

**NP**: “The class of dashed hopes and idle dreams.”\(^1\)

More formally:
the class of problems for which a possible solution can be verified in \(P\)

**Definition 7.2**
The class of languages that have polynomial-time verifiers is called **NP**.

In other words: **NP** is the class of all languages \(\mathcal{L}\) such that:

- for every \(w \in \mathcal{L}\), there is a **certificate** \(c_w \in \Sigma^*\), where
- the length of \(c_w\) is polynomial in the length of \(w\), and
- the language \(\{ (w \# c_w) | w \in \mathcal{L} \} \) is in \(P\)

\(^1\)https://complexityzoo.uwaterloo.ca/Complexity_Zoo:N#np
More Examples of Problems in \textbf{NP}

\textbf{Hamiltonian Path}

\begin{itemize}
  \item \textit{Input:} An undirected graph $G$
  \item \textit{Problem:} Is there a path in $G$ that contains each vertex exactly once?
\end{itemize}

\textbf{$k$-Clique}

\begin{itemize}
  \item \textit{Input:} An undirected graph $G$
  \item \textit{Problem:} Does $G$ contain a fully connected graph (clique) with $k$ vertices?
\end{itemize}
More Examples of Problems in \(\text{NP}\)

**SUBSET SUM**

Input: A collection of positive integers \(S = \{a_1, \ldots, a_k\}\) and a target integer \(t\).

Problem: Is there a subset \(T \subseteq S\) such that \(\sum_{a_i \in T} a_i = t\)?

**TRAVELLING SALESPERSON**

Input: A weighted graph \(G\) and a target number \(t\).

Problem: Is there a simple path in \(G\) with weight \(\leq t\)?
Complements of NP are often not known to be in NP

**No Hamiltonian Path**

*Input:* An undirected graph $G$

*Problem:* Is there no path in $G$ that contains each vertex exactly once?

Whereas it is easy to certify that a graph has a Hamiltonian path, there does not seem to be a polynomial certificate that it has not.

But we may just not be clever enough to find one.
More Examples

**Composite (non-prime) Number**

*Input:* A positive integer $n > 1$

*Problem:* Are there integers $u, v > 1$ such that $u \cdot v = n$?

**Prime Number**

*Input:* A positive integer $n > 1$

*Problem:* Is $n$ a prime number?

Surprisingly: both are in $\mathbb{NP}$ (see Wikipedia “Primality certificate”)

In fact: Composite Number (and thus Prime Number) was shown to be in $\mathbb{P}$
\( \mathbf{N} \) is for Nondeterministic
Reprise: Nondeterministic Turing Machines

A nondeterministic Turing Machine (NTM) \( \mathcal{M} = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}) \) consists of

- a finite set \( Q \) of states,
- an input alphabet \( \Sigma \) not containing \( \square \),
- a tape alphabet \( \Gamma \) such that \( \Gamma \supseteq \Sigma \cup \{ \square \} \),
- a transition function \( \delta : Q \times \Gamma \rightarrow \mathcal{P}(Q \times \Gamma \times \{L, R\}) \),
- an initial state \( q_0 \in Q \),
- an accepting state \( q_{\text{accept}} \in Q \).

Note

An NTM can halt in any state if there are no options to continue \( \leadsto \) no need for a special rejecting state
Reprise: Runs of NTMs

An (N)TM configuration can be written as a word $uqv$ where $q \in Q$ is a state and $uv \in \Gamma^*$ is the current tape contents.

NTMs produce configuration trees that contain all possible runs:

- **accept:**
  - $q_{\text{start}} \sigma_1 \cdots \sigma_n$
  - Non-deterministic choice
  - $q_{\text{acc}}$

- **reject:**
  - $q_{\text{start}} \sigma_1 \cdots \sigma_n$
  - Computation path
  - $\neq q_{\text{acc}}$

- **reject (not halting):**
  - $q_{\text{start}} \sigma_1 \cdots \sigma_n$
  - Infinite run
Example: Multi-Tape NTM

Consider the NTM \( M = (Q, \{0, 1\}, \{0, 1, \square\}, q_0, \Delta, q_{\text{accept}}) \) where

\[
\Delta = \left\{ 
\begin{array}{l}
(q_0, (\_), q_0, (0), (N)) \\
(q_0, (\_), q_0, (1), (N)) \\
(q_0, (\_), q_{\text{check}}, (\_), (N)) \\
\vdots \\
\text{transition rules for } M_{\text{check}} \\
\end{array}
\right\}
\]

and where \( M_{\text{check}} \) is a deterministic TM deciding whether number on second tape is \( > 1 \) and divides the number on the first.
Q: Which of the nondeterministic runs do time/space bounds apply to?
A: To all of them!

Definition 7.3
Let $M$ be a nondeterministic Turing machine and let $f : \mathbb{N} \to \mathbb{R}^+$ be a function.

- $M$ is $f$-time bounded if it halts on every input $w \in \Sigma^*$ and on every computation path after $\leq f(|w|)$ steps.
- $M$ is $f$-space bounded if it halts on every input $w \in \Sigma^*$ and on every computation path using $\leq f(|w|)$ cells on its tapes.

(Here we typically assume that Turing machines have a separate input tape that we do not count in measuring space complexity.)
Nondeterministic Complexity Classes

Definition 7.4

Let \( f : \mathbb{N} \to \mathbb{R}^+ \) be a function.

- \( \text{NTime}(f(n)) \) is the class of all languages \( \mathcal{L} \) for which there is an \( O(f(n)) \)-time bounded nondeterministic Turing machine deciding \( \mathcal{L} \), for some \( k \geq 1 \).

- \( \text{NSpace}(f(n)) \) is the class of all languages \( \mathcal{L} \) for which there is an \( O(f(n)) \)-space bounded nondeterministic Turing machine deciding \( \mathcal{L} \).
All Complexity Classes Have a Nondeterministic Variant

\[
\text{NPTIME} = \bigcup_{d \geq 1} \text{NTIME}(n^d) \quad \text{nondet. polynomial time}
\]

\[
\text{NEXP} = \text{NEXPTime} = \bigcup_{d \geq 1} \text{NTIME}(2^{n^d}) \quad \text{nondet. exponential time}
\]

\[
\text{N2EXP} = \text{N2EXPTime} = \bigcup_{d \geq 1} \text{NTIME}(2^{2^{n^d}}) \quad \text{nondet. double-exponential time}
\]

\[
\text{NL} = \text{NLogSpace} = \text{NSpace}(\log n) \quad \text{nondet. logarithmic space}
\]

\[
\text{NPSPACE} = \bigcup_{d \geq 1} \text{NSpace}(n^d) \quad \text{nondet. polynomial space}
\]

\[
\text{NEXPSPACE} = \bigcup_{d \geq 1} \text{NSpace}(2^{n^d}) \quad \text{nondet. exponential space}
\]
Equivalence of $\text{NP and NPTIME}$

**Theorem 7.5**
$\text{NP} = \text{NPTIME}$.

**Proof.**
- Suppose $L \in \text{NPTIME}$.
- Then there is an NTM $M$ such that
  \[ w \in L \iff \text{there is an accepting run of } M \text{ of length } O(n^d) \]
  for some $d$.
- This path can be used as a certificate for $w$.
- A DTM can check in polynomial time that a candidate certificate is a valid accepting run.

Therefore $\text{NP} \supseteq \text{NPTIME}$. 
Equivalence of $\text{NP}$ and $\text{NPTIME}$

Proof of the converse direction:

- Assume $L$ has a polynomial-time verifier $M$ with certificates of length at most $p(n)$ for a polynomial $p$.
- Then we can construct an NTM $M^*$ deciding $L$ as follows:
  1. $M^*$ guesses a string of length $p(n)$
  2. $M^*$ checks in deterministic polynomial time if this is a certificate.

Therefore $\text{NP} \subseteq \text{NPTIME}$. □
NP and coNP

Note: Definition of NP is not symmetric

- there does not seem to be any polynomial certificate for Sudoku unsolvability or propositional logic unsatisfiability . . .
- converse of an NP problem is coNP
- similar for \( \text{NExpTime} \) and \( \text{N2ExpTime} \)

Other complexity classes are symmetric:

- Deterministic classes (\( \text{coP} = \text{P} \) etc.)
- Space classes mentioned above (esp. \( \text{coNL} = \text{NL} \))
Deterministic vs. Nondeterministic Time

Theorem 7.6

\[ P \subseteq NP, \text{ and also } P \subseteq coNP. \]

(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?”
- Unresolved since over 30 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. $NP$

- It is often said: “Most experts think $P \neq NP$”
  - Main argument: “If $NP = P$, someone ought to have found some polynomial algorithm 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)

- Results of a poll among 100 experts [Gasarch 2002]:
  - $P \neq NP$: 61
  - $P = NP$: 9
  - No comment: 22
  - Other: independent (4), not independent (3), it depends (1)

- Over 100 “proofs” show $P = NP$ to be true/false/both/neither: https://www.win.tue.nl/~gwoegi/P-versus-NP.htm

- Many solutions conceivable, e.g., $P = NP$ could be shown with a non-constructive proof
A Simple Proof for $P = NP$

Clearly

therefore

hence

that is

using $coP = P$

and hence

so by $P \subseteq NP$  

$\mathcal{L} \in P$  implies  $\mathcal{L} \in NP$

$\mathcal{L} \notin NP$  implies  $\mathcal{L} \notin P$

$\mathcal{L} \in coNP$  implies  $\mathcal{L} \in coP$

$coNP \subseteq coP$

$coNP \subseteq P$

$NP \subseteq P$

$NP = P$

q.e.d.?