Theoretical Computer Science
Complexity Theory

Jonas Hübotter

Outline

$\mathcal{P}$

$\mathcal{NP}$

- Polynomially Bounded Verifier
- Polynomial Reduction
- $\mathcal{NP}$-completeness

Approximation Algorithm

$\mathcal{P}$

$\mathcal{P}$ is the class of problems that can be solved by a DTM in polynomial time.

Definition 1

We define

$$\begin{aligned} \mathrm{time}_M(w) &= (\#\text{steps until DTM } M[w] \text{ halts}) \in \mathbb{N} \cup \{\infty\} \\ \mathrm{TIME}(f(n)) &= \{A \subseteq \Sigma^* \mid \exists \text{DTM } M.\ A = L(M) \wedge \forall w \in \Sigma^*. \\ &\qquad \mathrm{time}_M(w) \leq f(|w|)\} \\ &\qquad \text{for a total function } f : \mathbb{N} \to \mathbb{N} \end{aligned}$$

Then, the complexity class $\mathcal{P}$ is given as

$$\mathcal{P} = \bigcup_{p \in \text{polynomial}} \mathrm{TIME}(p(n)) = \bigcup_{k \geq 0} \mathrm{TIME}(\mathcal{O}(n^k))$$

where $\mathrm{TIME}(\mathcal{O}(f)) = \bigcup_{g \in \mathcal{O}(f)} \mathrm{TIME}(g)$.

$\mathcal{NP}$

$\mathcal{NP}$ is the class of problems that can be solved by a NTM in polynomial time.

Definition 2

We define

$$\text{ntime}_M(w) = \begin{cases} (\text{minimal \#steps for NTM } M[w] \text{ to halt}) & w \in L(M) \\ 0 & w \notin L(M) \end{cases}$$

$$\begin{aligned} \text{NTIME}(f(n)) = \{A \subseteq \Sigma^* \mid \exists \text{NTM } M.\ A = L(M) \land \forall w \in \Sigma^*. \\ \text{ntime}_M(w) \leq f(|w|)\} \\ \text{for a total function } f : \mathbb{N} \to \mathbb{N} \end{aligned}$$

$\mathcal{NP}$

To simplify the notation, we do not require that the NTM M terminates for inputs $w \notin L(M)$. This is not a restriction as we can always define the NTM M' which returns 0 after $p(|w|)$ steps (timeout).

Definition 3

The complexity class $\mathcal{NP}$ is given as

$$\mathcal{NP} = \bigcup_{p \in \text{polynomial}} \mathsf{NTIME}(p(n)) = \bigcup_{k \geq 0} \mathsf{NTIME}(\mathcal{O}(n^k)).$$

Polynomially Bounded Verifier

A problem A is in $\mathcal{NP}$ if and only if solutions (which are described by *certificates*) to the problem can be verified in polynomial time by a DTM (a *polynomially bounded verifier*).

Intuition

- A decision problem can be thought of an exploration of the search space consisting of all instances with the goal of finding a solution.
- Problems in $\mathcal{NP}$ may be harder than problems in $\mathcal{P}$ as a NTM is able to pursue exponentially many paths in the search tree.
- However, once the NTM found a path, the length of this path must be polynomial in the size of the input.
- Thus, a DTM must be able to verify that a given path is correct in polynomial time.

Polynomially Bounded Verifier

Definition 4

Let M be a DTM with $L(M) = \{w\#c \mid w \in \Sigma^*, c \in \Delta^*\}$.

- If $w\#c \in L(M)$, c is a certificate for w.
- M is a polynomially bounded verifier for the language $\{w \in \Sigma^* \mid \exists c \in \Delta^*.\ w\#c \in L(M)\}$ (i.e. the language of all words that have a certificate) if there exists a polynomial p such that $\text{time}_M(w\#c) \leq p(|w|)$.

Especially: $c \leq p(|w|)$, i.e. the size of the certificate must be polynomially bounded by the size of the input.

Polynomial Reduction

Definition 5

Given problems $A \subseteq \Sigma^*, B \subseteq \Gamma^*$, A is polynomially reducable to B (denoted $A \leq_p B$) if there exists a total and by a DTM in polynomial time computable function $f : \Sigma^* \to \Gamma^*$ such that

$$\forall w \in \Sigma^*.\ w \in A \iff f(w) \in B.$$

The complexity classes $\mathcal{P}$ and $\mathcal{NP}$ are closed under polynomial reduction.

$\mathcal{NP}$-completeness

Definition 6

- The language L is $\mathcal{NP}$-hard if $\forall A \in \mathcal{NP}.\ A \leq_p L$.
- The language L is $\mathcal{NP}$-complete if $L \in \mathcal{NP}$ and L is $\mathcal{NP}$-hard.

Example 7 (Proving $\mathcal{NP}$-completeness)

- If $A \leq_p B$ and A is $\mathcal{NP}$-hard, then B is $\mathcal{NP}$-hard.
- If $A \leq_p B$ and $B \in \mathcal{NP}$, then $A \in \mathcal{NP}$.
- If there exists a polynomially bounded verifier for A, then $A \in \mathcal{NP}$.

Approximation Algorithm

Definition 8

A *d-approximation algorithm* ($d \in \mathbb{R}$) for an optimization problem is an algorithm that computes in polynomial time a solution to the problem that is at most d times worse than the optimal solution.