

Complexity Theory

Lecture 8

Anuj Dawar

<http://www.cl.cam.ac.uk/teaching/1920/Complexity>

Responses to NP-Completeness

Confronted by an NP-complete problem, say constructing a timetable, what can one do?

- It's a single instance, does asymptotic complexity matter?
- What's the critical size? Is scalability important?
- Are there guaranteed restrictions on the input? Will a special purpose algorithm suffice?
- Will an approximate solution suffice? Are performance guarantees required?
- Are there useful heuristics that can constrain a search? Ways of ordering choices to control backtracking?
- Can you use a SAT-solver?

Validity

We define **VAL**—the set of *valid* Boolean expressions—to be those Boolean expressions for which every assignment of truth values to variables yields an expression equivalent to **true**.

$$\phi \in \text{VAL} \iff \neg\phi \notin \text{SAT}$$

By an exhaustive search algorithm similar to the one for **SAT**, **VAL** is in $\text{TIME}(n^2 2^n)$.

Is **VAL** \in **NP**?

Validity

$\overline{\text{VAL}} = \{\phi \mid \phi \notin \text{VAL}\}$ —the *complement* of VAL is in NP.

Guess a *falsifying* truth assignment and verify it.

Such an algorithm does not work for VAL.

In this case, we have to determine whether *every* truth assignment results in *true*—a requirement that does not sit as well with the definition of acceptance by a nondeterministic machine.

Complementation

If we interchange accepting and rejecting states in a deterministic machine that decides the language L , we get one that accepts \bar{L} .

If a language $L \in P$, then also $\bar{L} \in P$.

Complexity classes defined in terms of nondeterministic machine models are not necessarily closed under complementation of languages.

Define,

co-NP – the languages whose complements are in **NP**.

Succinct Certificates

The complexity class NP can be characterised as the collection of languages of the form:

$$L = \{x \mid \exists yR(x, y)\}$$

Where R is a relation on strings satisfying two key conditions

1. R is decidable in polynomial time.
2. R is *polynomially balanced*. That is, there is a polynomial p such that if $R(x, y)$ and the length of x is n , then the length of y is no more than $p(n)$.

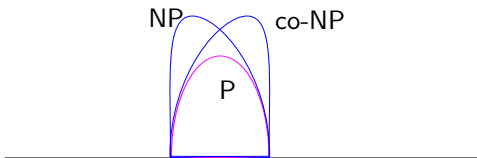
co-NP

As **co-NP** is the collection of complements of languages in **NP**, and **P** is closed under complementation, **co-NP** can also be characterised as the collection of languages of the form:

$$L = \{x \mid \forall y \mid y \mid < p(|x|) \rightarrow R'(x, y)\}$$

NP – the collection of languages with succinct certificates of membership.

co-NP – the collection of languages with succinct certificates of disqualification.



Any of the situations is consistent with our present state of knowledge:

- $P = NP = \text{co-NP}$
- $P = NP \cap \text{co-NP} \neq NP \neq \text{co-NP}$
- $P \neq NP \cap \text{co-NP} = NP = \text{co-NP}$
- $P \neq NP \cap \text{co-NP} \neq NP \neq \text{co-NP}$

co-NP-complete

VAL – the collection of Boolean expressions that are *valid* is *co-NP-complete*.

Any language L that is the complement of an NP-complete language is *co-NP-complete*.

Any reduction of a language L_1 to L_2 is also a reduction of \bar{L}_1 —the complement of L_1 —to \bar{L}_2 —the complement of L_2 .

There is an easy reduction from the complement of **SAT** to **VAL**, namely the map that takes an expression to its negation.

$$\text{VAL} \in \text{P} \Rightarrow \text{P} = \text{NP} = \text{co-NP}$$

$$\text{VAL} \in \text{NP} \Rightarrow \text{NP} = \text{co-NP}$$

Prime Numbers

Consider the decision problem **PRIME**:

Given a number x , is it prime?

This problem is in **co-NP**.

$$\forall y (y < x \rightarrow (y = 1 \vee \neg(\text{div}(y, x))))$$

Note again, the algorithm that checks for all numbers up to \sqrt{n} whether any of them divides n , is not polynomial, as \sqrt{n} is not polynomial in the size of the input string, which is $\log n$.

Primality

Another way of putting this is that **Composite** is in **NP**.

Pratt (1976) showed that **PRIME** is in **NP**, by exhibiting succinct certificates of primality based on:

*A number $p > 2$ is **prime** if, and only if, there is a number r , $1 < r < p$, such that $r^{p-1} = 1 \pmod p$ and $r^{\frac{p-1}{q}} \neq 1 \pmod p$ for all **prime divisors** q of $p - 1$.*

Primality

In 2002, Agrawal, Kayal and Saxena showed that PRIME is in P.

If a is co-prime to p ,

$$(x - a)^p \equiv (x^p - a) \pmod{p}$$

if, and only if, p is a prime.

Checking this equivalence would take too long. Instead, the equivalence is checked *modulo* a polynomial $x^r - 1$, for “suitable” r .

The existence of suitable small r relies on deep results in number theory.

Factors

Consider the language **Factor**

$$\{(x, k) \mid x \text{ has a factor } y \text{ with } 1 < y < k\}$$

Factor \in NP \cap co-NP

Certificate of membership—a factor of x less than k .

Certificate of disqualification—the prime factorisation of x .

Graph Isomorphism

Given two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, is there a *bijection*

$$\iota : V_1 \rightarrow V_2$$

such that for every $u, v \in V_1$,

$$(u, v) \in E_1 \quad \text{if, and only if,} \quad (\iota(u), \iota(v)) \in E_2.$$

Graph Isomorphism

Graph Isomorphism is

- in NP
- not known to be in P
- not known to be in co-NP
- not known (or *expected*) to be NP-complete
- recently shown to be in *quasi-polynomial time*, i.e. in

$$\text{TIME}(n^{(\log n)^k})$$

for a constant k .