Complexity Theory
Lecture 8

Anuj Dawar

University of Cambridge Computer Laboratory
Easter Term 2013

http://www.cl.cam.ac.uk/teaching/1213/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?

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`.

$$\phi \in \mathsf{VAL} \quad \Leftrightarrow \quad \neg\phi \notin \mathsf{SAT}$$

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

Is $\mathsf{VAL} \in \mathsf{NP}$?

Validity

$\overline{\mathsf{VAL}} = \{\phi \mid \phi \notin \mathsf{VAL}\}$—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 accepts the language L, we get one that accepts $\overline{L}$.

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

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:

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

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 $p(n)$.

Succinct Certificates

y is a *certificate* for the membership of x in L.

Example: If L is SAT, then for a satisfiable expression x, a certificate would be a satisfying truth assignment.

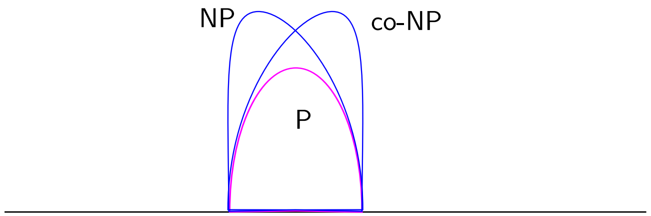
co-NP

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:

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

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:

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

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.

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

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

Prime Numbers

Consider the decision problem PRIME:

Given a number x, is it prime?

This problem is in co-NP.

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

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 $r^{p-1} = 1 \bmod p$ and $r^{\frac{p-1}{q}} \neq 1 \bmod p$ 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 to 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.