

# A Taste of Complexity

Peter Brooksbank

Department of Mathematics

Bucknell University

`pbrooksb@bucknell.edu`

Student Colloquium Series

Bucknell University

Thursday 23rd February, 2006

## Examples of Computational Problems

**prime** Given an integer  $N$ , decide whether  $N$  is prime.

**factorise** Given an integer  $N$ , write  $N = p_1 \cdot p_2 \cdot \dots \cdot p_k$ , where each  $p_i$  is prime.

**salesman** Given the cost of traveling between each pair of  $n$  cities, find a tour of the cities of minimal cost (beginning and ending at the same city).

# Algorithms

By a **solution** to a computational problem, we mean an **algorithm** that takes, as input, any permissible input  $I$  to the problem, and outputs an answer.

In this talk we will only be concerned with

- **Deterministic algorithms** guaranteed to terminate with a correct solution for any permissible input.
- **Decision problems** problems for which there are only two possible correct outputs: “**yes**” or “**no**”

## Examples of Decision Problems

**factor** Given integers  $N$  and  $d$ , decide whether  $d$  divides  $N$ .

**prime** and its close relative...

**composite** Given an integer  $N$ , decide whether  $N$  is composite.

we can also convert problems to decision problems...

**salesman** Given an input to original problem and an integer  $M$ , decide whether there is a tour of the cities costing at most  $M$ .

# Examples of Algorithms

**factor** use the **division algorithm** to write  $N = qd + r$ , with  $0 \leq r < d$ ; return “yes” if and only if  $r = 0$ .

**salesman** list all  $\frac{n!}{2}$  tours and see if any cost at most  $M$ .

**prime**

for each  $d$  in  $\{2, \dots, \sqrt{N}\}$  do

    if **factor**( $N$ ,  $d$ ) then

        return “no”

return “yes”

**Question:** Do you think the algorithm for **salesman** is a “good” algorithm? What about the algorithm for **prime**?

# Algorithmic Efficiency

We typically measure the efficiency of an algorithm by comparing the number of basic computations performed by the algorithm with the **length** of the input.

prime  $|I| = \log N$     #steps =  $\sqrt{N} = 2^{|I|/2}$

salesman  $|I| \approx n^2$     #steps  $\approx n! > 2^{|I|/2}$

The number of basic steps used by both algorithms is **exponential** in the length of an input.

They should therefore not generally be regarded as “good”.

# Polynomial Time

An algorithm runs in **polynomial time** if there exists a polynomial  $p: \mathbb{N} \rightarrow \mathbb{N}$  such that, for any permissible input  $I$ , the number of basic steps performed by the algorithm is at most  $p(|I|)$ .

**Complexity** A decision problem is said to lie in the complexity class **P** if there exists a polynomial time algorithm that solves the problem. (Example: **factor** is in **P**)

## Questions:

- Is **prime** in **P**? (Equivalently, is **composite** in **P**?)
- Is **salesman** in **P**?

We generally regard problems that lie in **P** as being “easy”.

# Non-deterministic polynomial time

**composite** has the following property:

if  $N$  is composite, one can provide a “witness” to this fact, namely an integer  $d$  that divides  $N$

AND the fact that  $d$  divides  $N$  can be verified in polynomial time.

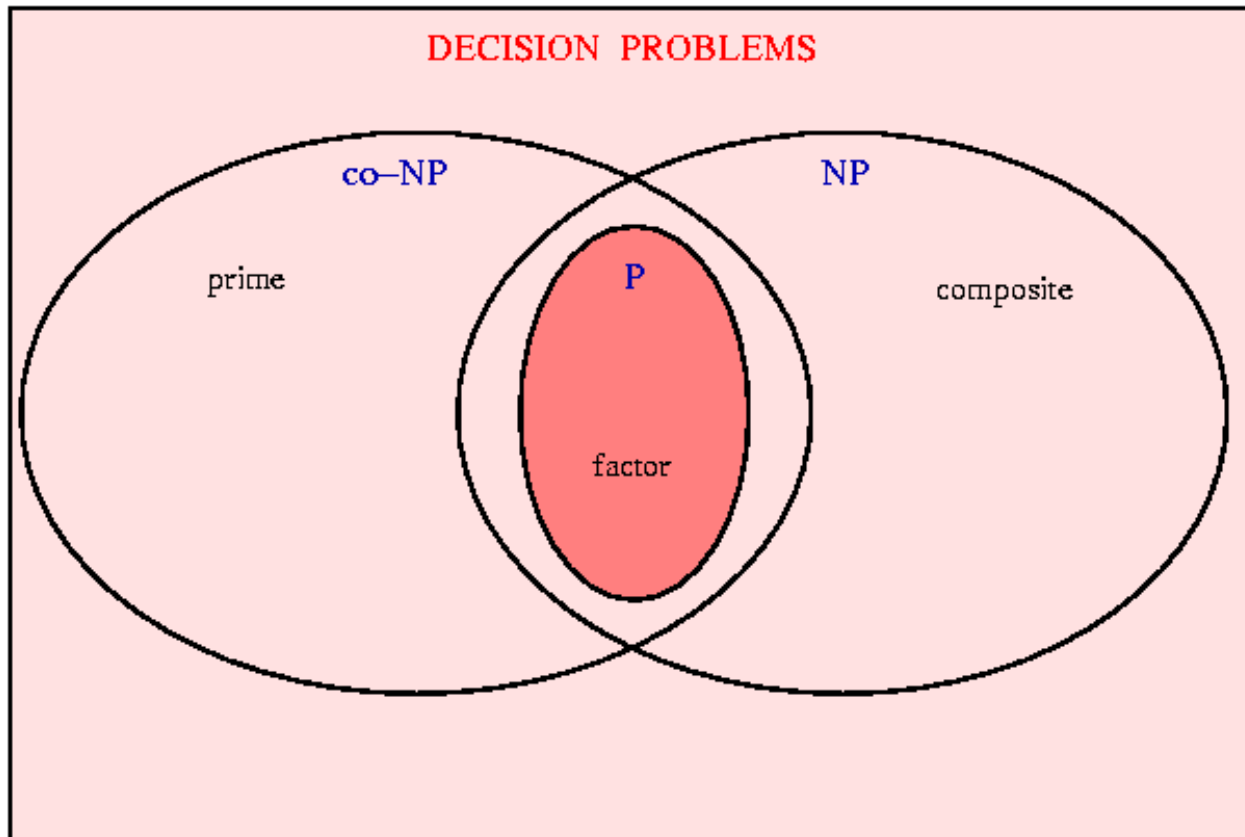
A decision problem is in the complexity class **NP** if, whenever the correct answer is “yes” for some input  $I$ , a “witness” or “certificate” can be provided that can be verified in polynomial time.

(Note that all decision problems in **P** are also in **NP**)

**composite** is in **NP**

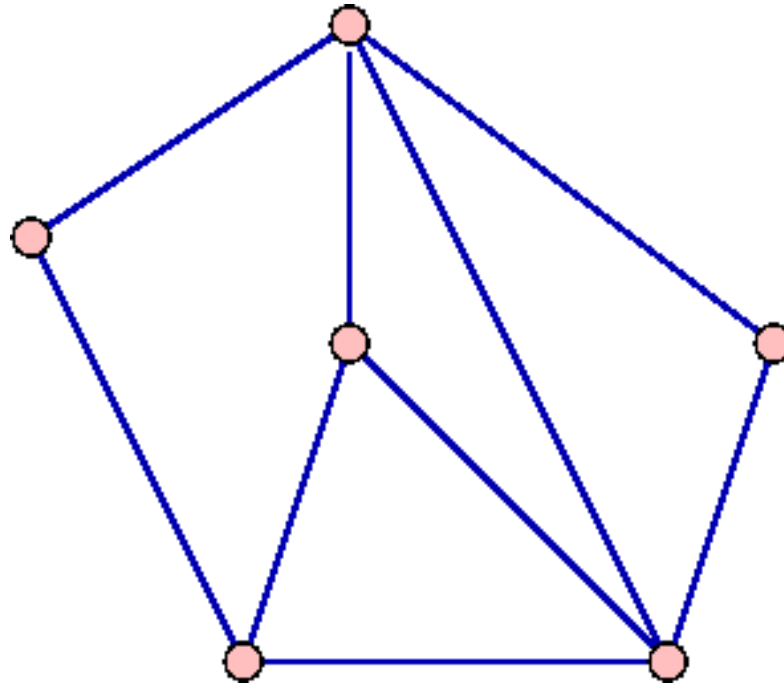
**Question:** Is **prime** in **NP**?

# Map of Complexity



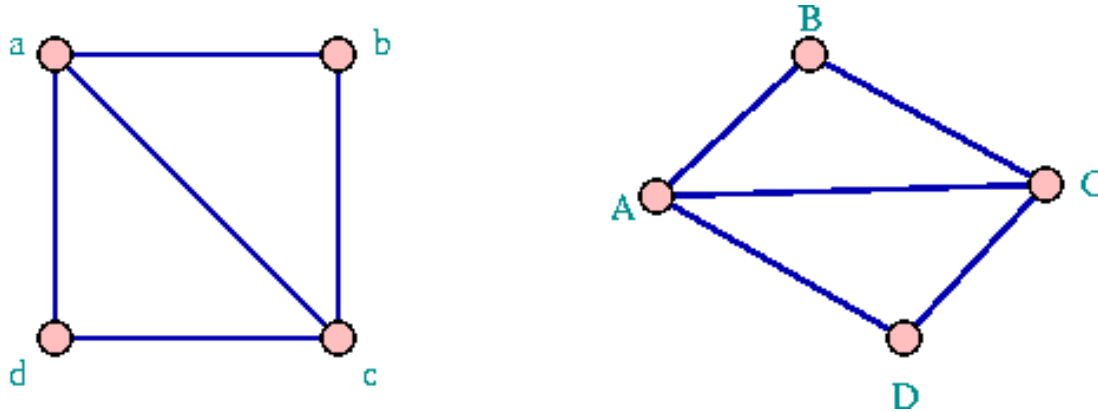
# Graph Theory: A Source of Interesting Problems

A graph  $G$  consists of a set  $V(G)$  of vertices and a set  $E(G)$  of edges connecting these vertices.



# Graph Isomorphism

Graphs  $G$  and  $H$  are **isomorphic** if there is a bijection  $V(G) \rightarrow V(H)$  that induces a bijection  $E(G) \rightarrow E(H)$



The map  $a \mapsto A$ ,  $b \mapsto B$ ,  $c \mapsto C$ ,  $d \mapsto D$  is an isomorphism.

**graph iso** Decide whether two given graphs are isomorphic.

Clearly **graph iso** is in **NP**

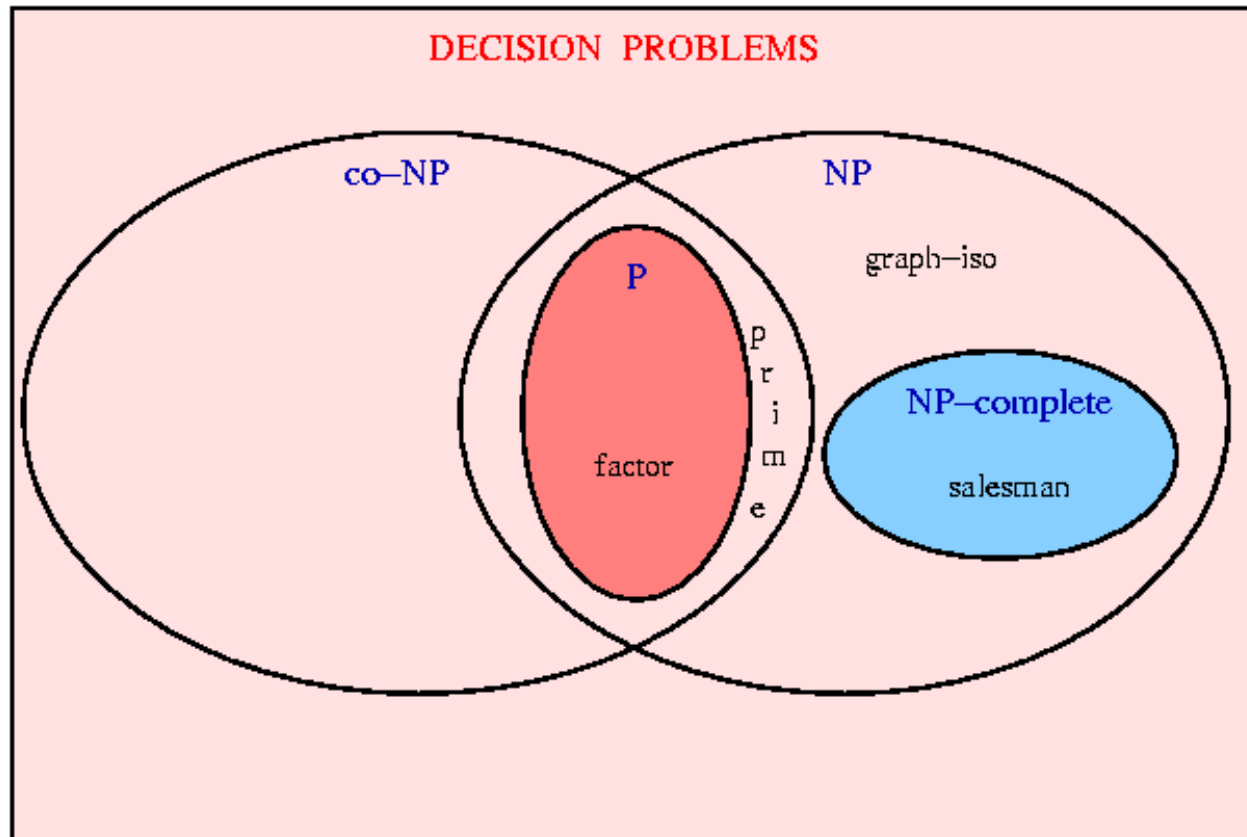
**Question:** Is **graph iso** in **P**?

## NP-completeness

Within **NP** is a subclass of the “hardest” problems. More precisely, a problem is said to be **NP-complete** if any algorithm to solve the problem can be adapted (in polynomial time) to solve **ANY** other **NP** problem.

**Question:** Are there any **NP-complete** problems?

# Map of Complexity



# Undecidability

What lies outside  $NP \cup \text{co-NP}$ ?

**halt** Given a description of a program  $P$  and a permissible input  $I$  to  $P$ , decide whether  $P$  terminates with input  $I$ .

**Theorem (Turing, Church)** There is no algorithm (polynomial or otherwise) to solve **halt**.

We say that **halt** is **undecidable**.

## Illustration: Twin Primes

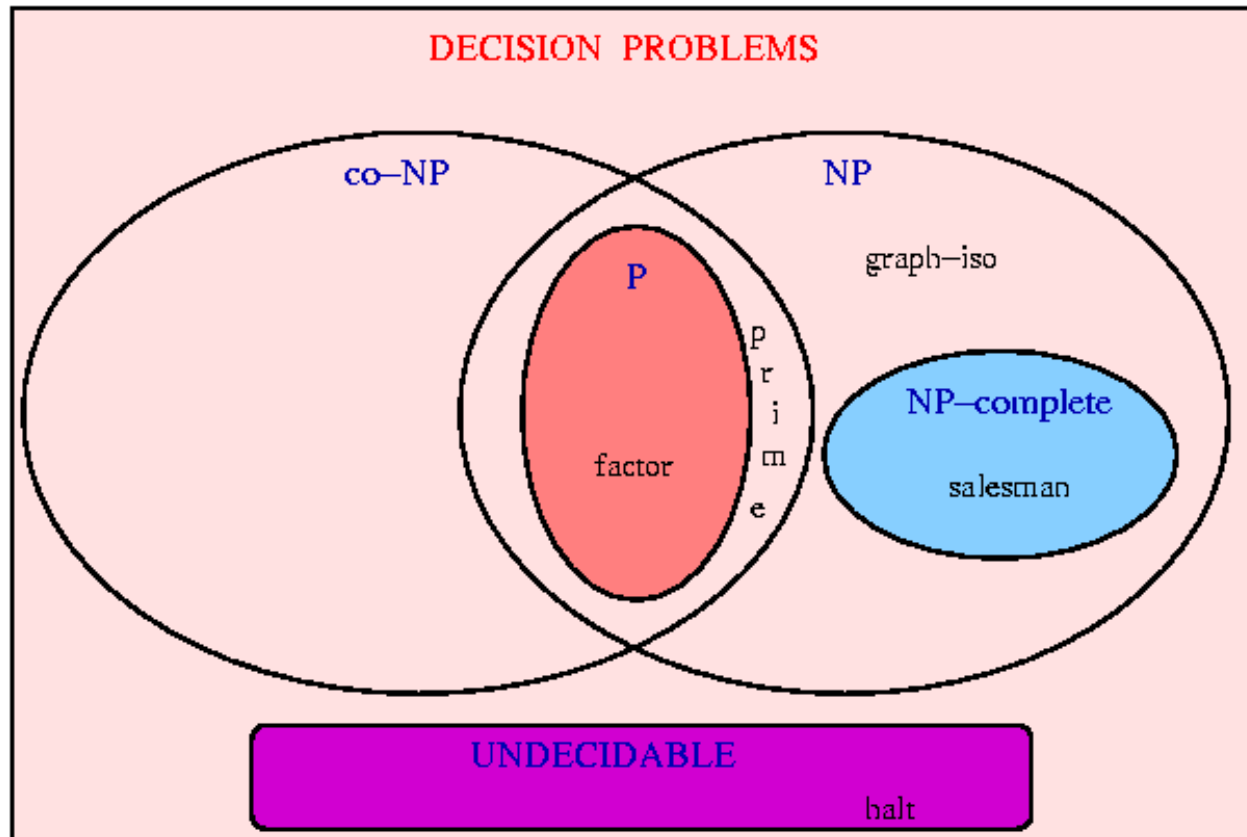
Consider the following program  $P$  with integer input  $N$

```
 $a := N$   
while  $\text{composite}(a)$  or  $\text{composite}(a + 2)$  do  
     $a := a + 1$   
return  $a, a + 2$ 
```

If  $P$  terminates for all inputs, then there are infinitely many **twin primes**; if not, then there is largest pair.

Thus any algorithm that can decide whether  $P$  halts for all possible inputs would, in principle, provide an answer to the long standing **twin prime conjecture**.

# Map of Complexity



# A Millennium Problem

$$P \neq NP?$$

In a recent poll of 100 researchers:

- 61 said “yes”
- 9 said “no”
- 22 were unsure
- 8 believed that it is independent of currently accepted axioms

The person clever enough to resolve the question will earn a cool  
\$1,000,000.

## Possibilities

In 1974, Richard Karp suggested three decision problems that may reside in the strange world between  $P$  and  $NP$ -complete.

Since that time, two of the three have been shown to be in  $P$ .

The complexity status of the third is still unknown; this problem is  $graph\ iso$ .

# Map of Complexity

