The Complexity Classes P and NP

Andreas Klappenecker

[partially based on slides by Professor Welch]

P

Polynomial Time Algorithms

Most of the algorithms we have seen so far run in time that is upper bounded by a polynomial in the input size

- sorting: $O(n^2)$, $O(n \log n)$, ...
- matrix multiplication: $O(n^3)$, $O(n^{\log_2 7})$
- graph algorithms: $O(V+E)$, $O(E \log V)$, ...

In fact, the running time of these algorithms are bounded by small polynomials.

Categorization of Problems

We will consider a computational problem tractable if and only if it can be solved in polynomial time.

Decision Problems and the class P

A computational problem with yes/no answer is called a decision problem.

We shall denote by P the class of all decision problems that are solvable in polynomial time.

Why Polynomial Time?

It is convenient to define decision problems to be tractable if they belong to the class P, since

- the class P is closed under composition.

- the class P is nearly independent of the computational model.

[Of course, no one will consider a problem requiring an $\Omega(n^{100})$ algorithm as efficiently solvable. However, it seems that most problems in P that are interesting in practice can be solved fairly efficiently.]

NP

Efficient Certification

An efficient certifier for a decision problem X is a

- polynomial time algorithm that takes two inputs, an putative instance i of X and a certificate c and returns either yes or no.
- there is a polynomial such that for every string i, the string i is an instance of X if and only if there exists a string c such that c <= p(|i|) and B(d,c) = yes.

The certifier does not produce a solution to the decision problem X, but it is verifying that i belongs to X given a correct, short certificate c.

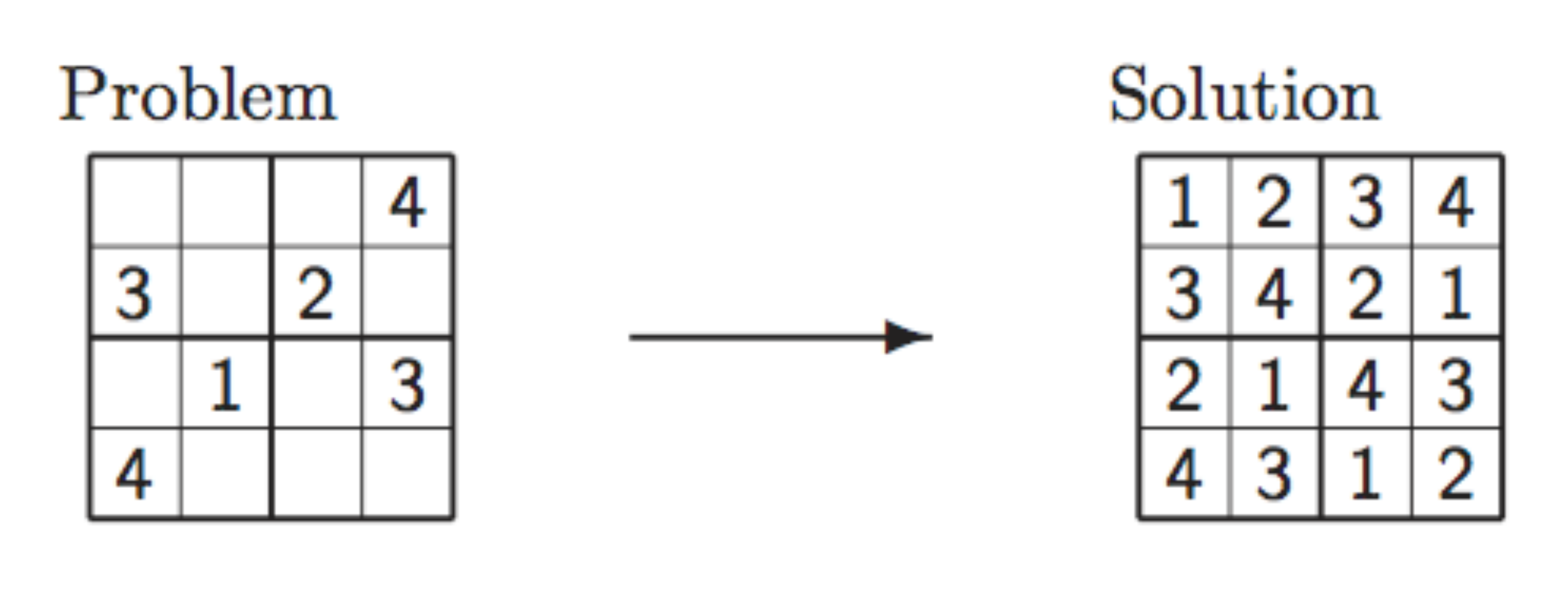
NP

The set of all decision problems that have an efficient certifier is called NP.

Sudoku

- The problem is given as an $n^2 \times n^2$ array which is divided into blocks of $n \times n$ squares.
- Some array entries are filled with an integer in the range $[1..\ n^2]$.
- The goal is to complete the array such that each row, column, and block contains each integer from $[1..n^2]$.

Sudoku



- Finding the solution might be difficult, but verifying the solution is easy.
- The Sudoku decision problem is whether a given Sudoku problem has a solution.

Hamiltonian Cycle

- A Hamiltonian cycle in an undirected graph is a cycle that visits every node exactly once.
- Solving a problem: Is there a Hamiltonian cycle in graph G?
- Verifying a candidate solution: Is $v_0, v_1, \ldots, v_\ell$ a Hamiltonian cycle of graph G?

Solving vs. Verifying

- Intuitively it seems much harder (more time consuming) in some cases to solve a problem from scratch than to verify that a candidate solution actually solves the problem.
- If there are many candidate solutions to check, then even if each individual one is quick to check, overall it can take a long time

The Class NP

NP is short for nondeterministic polynomial time, since the decision problem in NP are precisely the problems that can be solved on a nondeterministic Turing machine in polynomial time.

NP does not stand for "not P", as there are many problems that cannot even be verified in polynomial time.

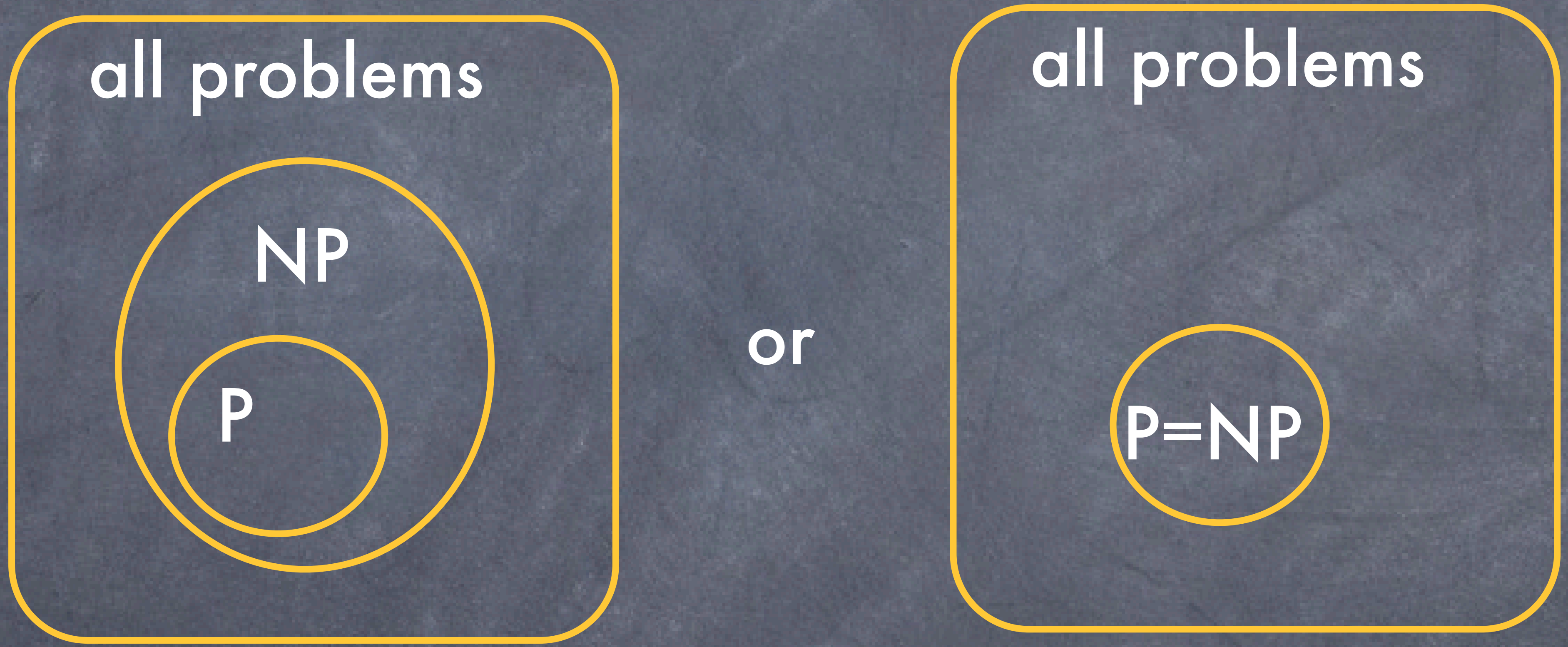
P versus NP

P vs. NP

- Although poly-time verifiability seems like a weaker condition than poly time solvability, no one has been able to prove that it is describes a larger class of problems.

- So it is unknown whether P = NP.

P and NP

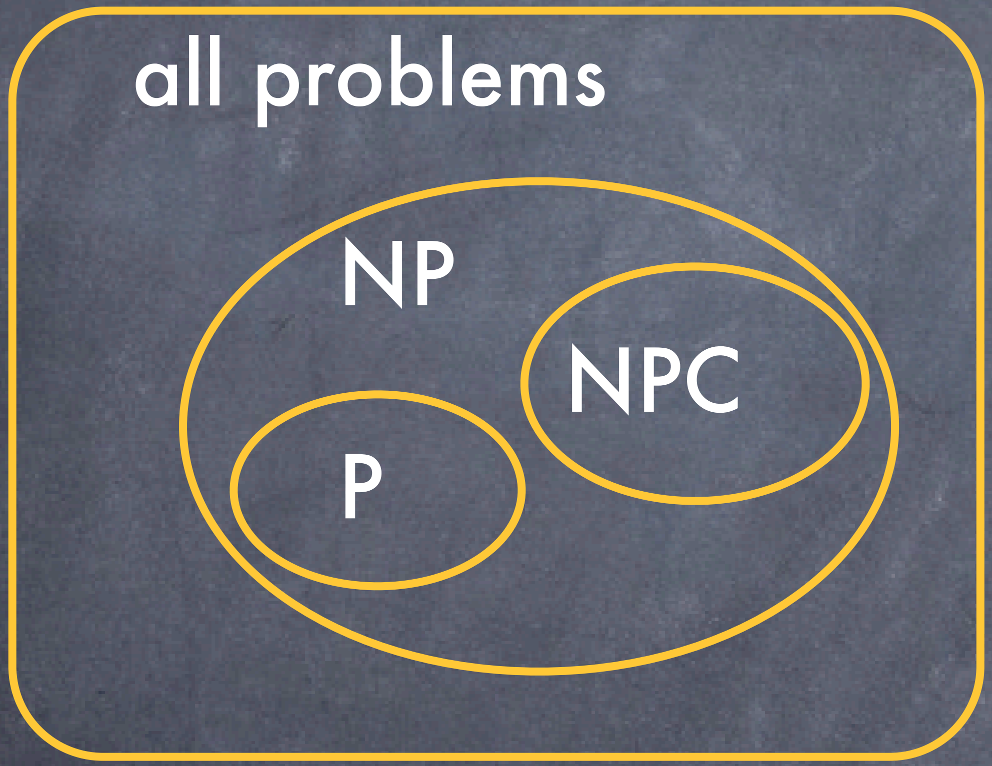


NP-Complete Problems

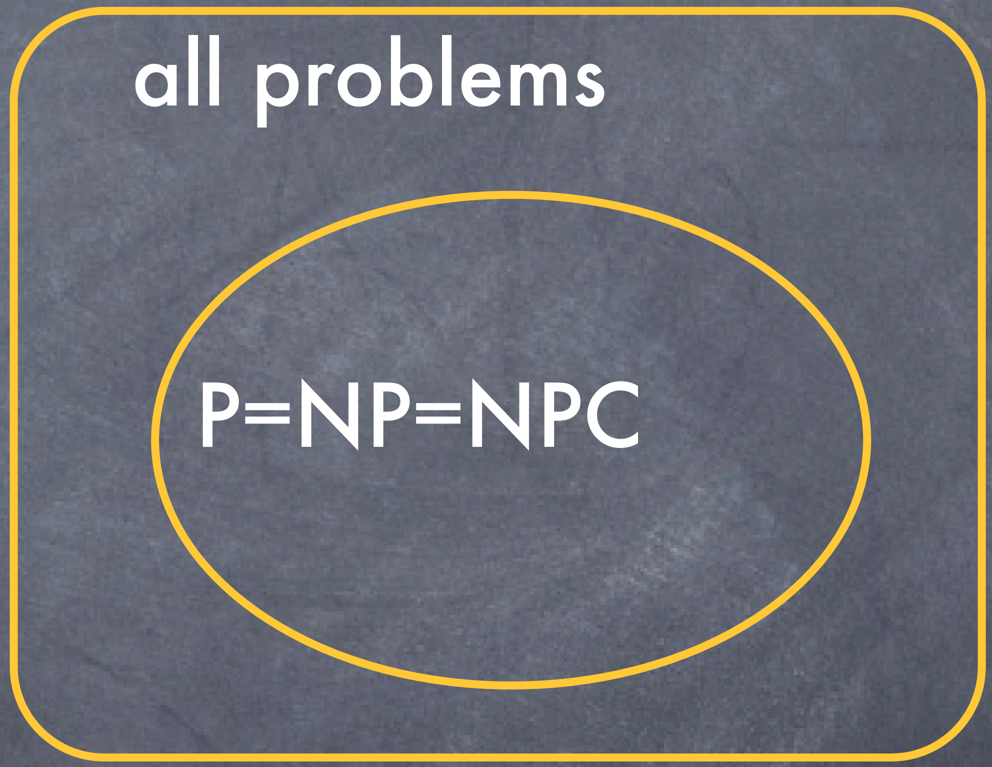
- NP-complete problems is class of "hardest" problems in NP.

- If an NP-complete problem can be solved in polynomial time, then all problems in NP can be solve in polynomial time, and thus P = NP.

Possible Worlds



or



NPC = NP-complete

P = NP Question

The question whether P=NP is open since the 70s. It is one of the central open problems in computer science. The question is of theoretical interest as well as of great practical importance.

Pragmatic approach: If your problem is NP-complete, then don't waste time looking for an efficient algorithm, focus on approximations or heuristics.