Title: Dynamical Systems - Review (PHYS 607) - Lecture 4

Date: Jan 07, 2010 09:00 AM

URL: http://pirsa.org/10010014

Abstract:

Some notes on computational complexity

S. N. Coppersmith
Department of Physics
University of Wisconsin

Computational complexity: study of how computational resources needed to solve a problem grow with size of problem specification

Relation to study of complex systems:

- Gives insight into resources needed to compute properties of a given model (e.g., spin glasses)
- Classic problems exhibit phase transitions with similarities to those observed in condensed matter systems

The complexity classes P and NP

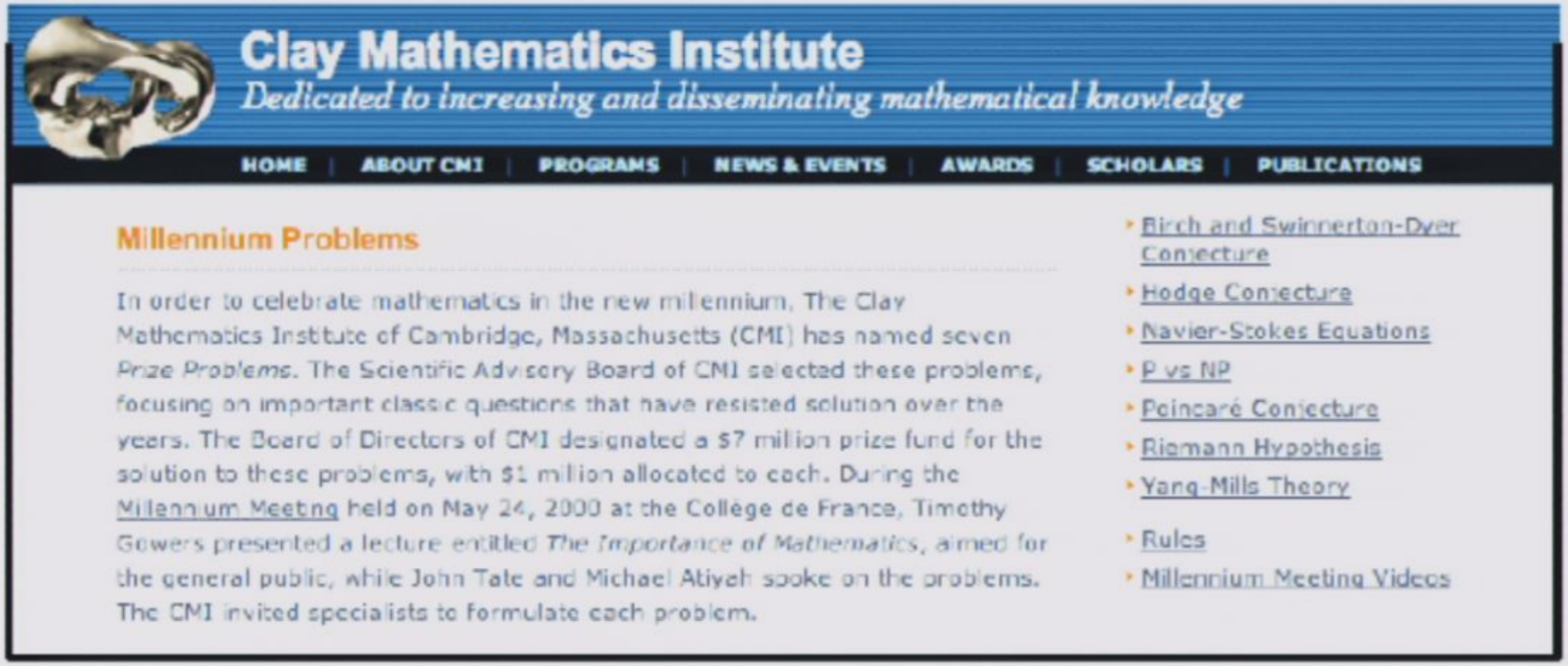
P: Problems that can be solved in a number of steps that grows no faster than polynomially with the size of the problem specification

NP: Problems for which a solution can be verified in a number of steps that grows no faster than polynomially with the size of the problem specification. (NP = nondeterministic polynomial)

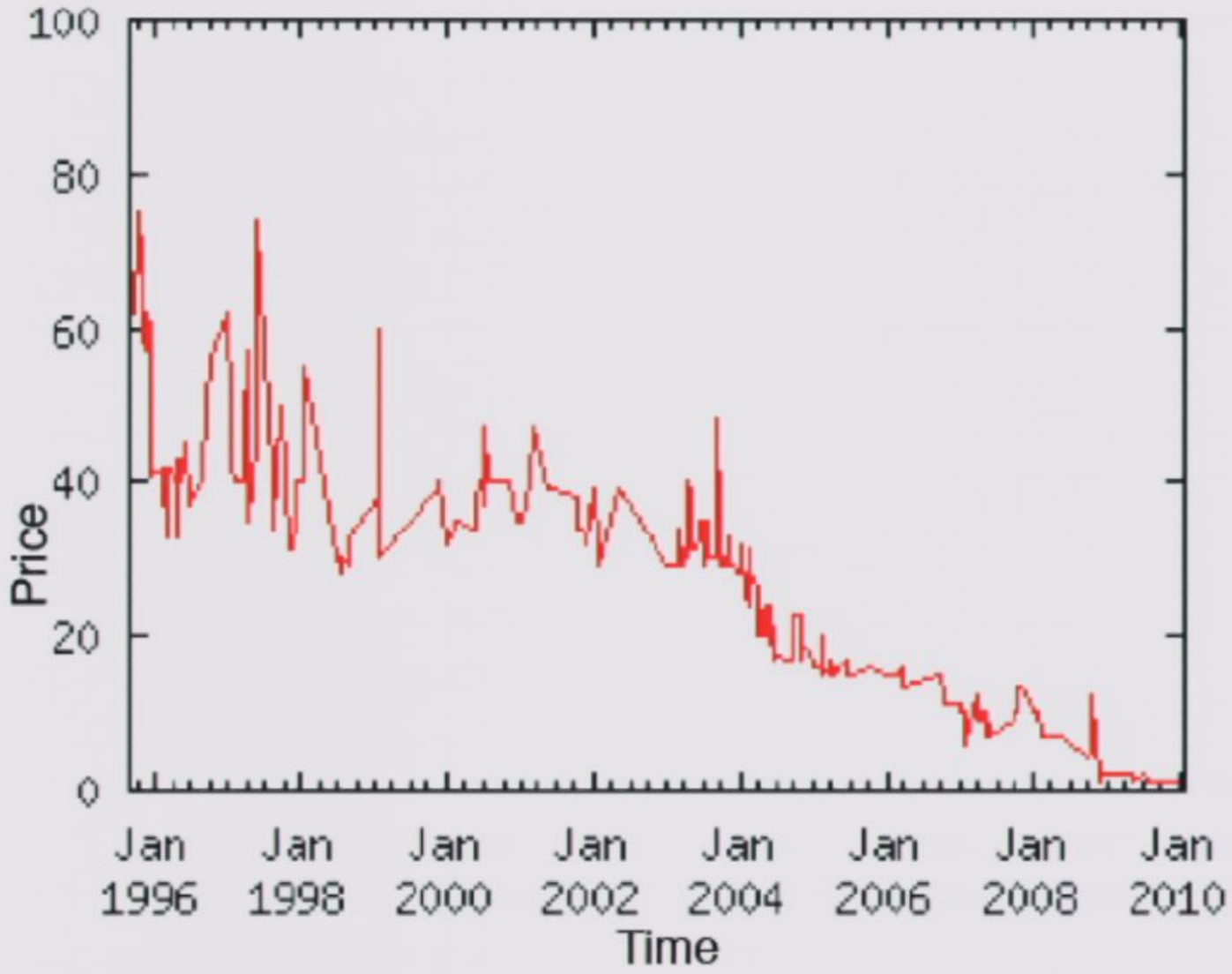
We know that problems in P are easy to solve.

We think that some problems in NP are hard to solve.

Whether or not P is distinct from NP is a key unanswered question in computational complexity



Market for P=NP or P≠NP proven by 2010



An example: satisfiability (SAT) - a classic problem in computational complexity

K-SAT:
Given N Boolean variables, and M clauses, each a disjunction (OR) of K literals (conditions that a variable is TRUE or FALSE), is there an assignment of the variables for which all of the clauses are true?

Example: the 3-SAT expression

$(x_1 \text{ OR } (\text{not } x_2) \text{ OR } x_3) \text{ AND } ((\text{not } x_1) \text{ OR } x_2 \text{ OR } (\text{not } x_3))$

is satisfiable, since it holds for the assignment
$x_1 = \text{TRUE}, x_2 = \text{TRUE}, x_3 = \text{FALSE}.$

Satisfiability example

At rock concert, each person in audience has a list of songs that he/she does or does not want played

(e.g., “Start Me Up” or “Brown Sugar or not “Angie”)

Can the songs in the concert be chosen so that every person in the audience has at least one satisfied request?

An example: satisfiability (SAT) - a classic problem in computational complexity

K-SAT:
Given N Boolean variables, and M clauses, each a disjunction (OR) of K literals (conditions that a variable is TRUE or FALSE), is there an assignment of the variables for which all of the clauses are true?

Example: the 3-SAT expression

$(x_1 \text{ OR } (\text{not } x_2) \text{ OR } x_3) \text{ AND } ((\text{not } x_1) \text{ OR } x_2 \text{ OR } (\text{not } x_3))$

is satisfiable, since it holds for the assignment
$x_1 = \text{TRUE}, x_2 = \text{TRUE}, x_3 = \text{FALSE}.$

NP-completeness

NP-complete problems are the problems in NP that we think are hold to solve. If a polynomial algorithm for solving an NP-complete problem exists, then every problem in NP can be solved in polynomial time.

3-SAT is first problem shown to be NP-complete (Cook, 1971).

An example: satisfiability (SAT) - a classic problem in computational complexity

K-SAT:
Given N Boolean variables, and M clauses, each a disjunction (OR) of K literals (conditions that a variable is TRUE or FALSE), is there an assignment of the variables for which all of the clauses are true?

Example: the 3-SAT expression

$(x_1 \text{ OR } (\text{not } x_2) \text{ OR } x_3) \text{ AND } ((\text{not } x_1) \text{ OR } x_2 \text{ OR } (\text{not } x_3))$

is satisfiable, since it holds for the assignment
$x_1 = \text{TRUE}, x_2 = \text{TRUE}, x_3 = \text{FALSE}.$

Satisfiability example

At rock concert, each person in audience has a list of songs that he/she does or does not want played

(e.g., “Start Me Up” or “Brown Sugar or not “Angie”)

Can the songs in the concert be chosen so that every person in the audience has at least one satisfied request?

NP-completeness

NP-complete problems are the problems in NP that we think are hold to solve. If a polynomial algorithm for solving an NP-complete problem exists, then every problem in NP can be solved in polynomial time.

3-SAT is first problem shown to be NP-complete (Cook, 1971).

Other examples of NP-complete problems:

Traveling salesman problem
Number partitioning
Ground state of 3-d Ising spin glass
Ground state of 2-d Ising spin glass in a magnetic field
Minimum vertex cover
Maximum clique
.
.
.

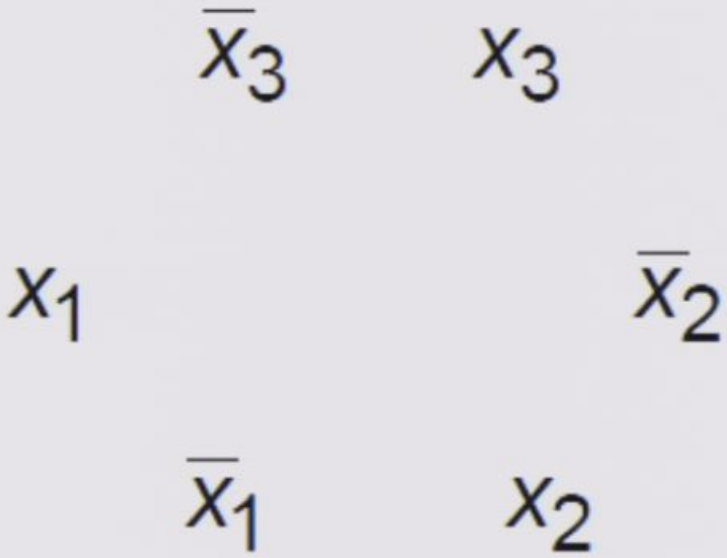
More than 3000 NP-complete problems are known.

How to show 3-SAT is NP-complete

1. Show SAT is in NP — for a problem with N literals, K literals per clause, and M clauses, satisfying assignment can be verified in polynomial time.
2. Show that computational steps to verify solution to any problem in NP can be written as a SAT expression (size is polynomial by definition of NP)
3. Convert N-SAT to 3-SAT by noting that (A or B or C or D) is satisfiable
   if and only if
   ((A or B or X) and ((not X) or C or D)) is

2-SAT is in P

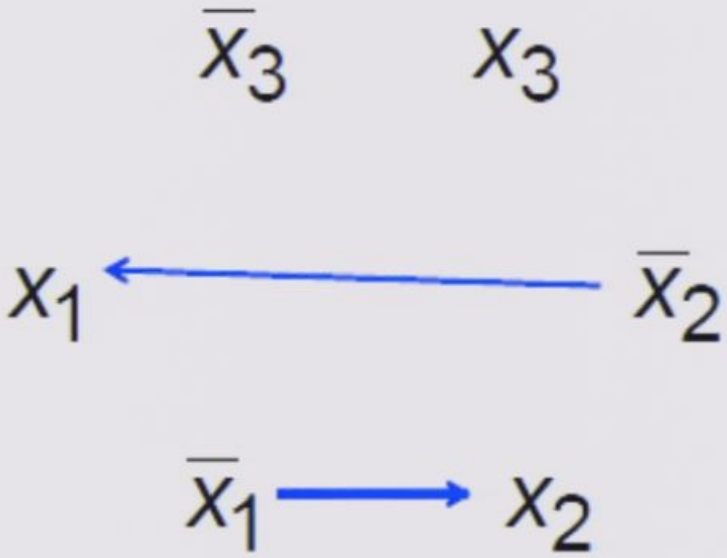
Linear time algorithm for 2-SAT: turn implications into links of a graph. Instance is unsatisfiable if there is path from x_i to (not x_i) and from (not x_i) to x_i for some i, and satisfiable otherwise



2-SAT is in P

Linear time algorithm for 2-SAT: turn implications into links of a graph. Instance is unsatisfiable if there is path from x_i to (not x_i) and from (not x_i) to x_i for some i, and satisfiable otherwise

$(x_1 \text{ OR } x_2)$

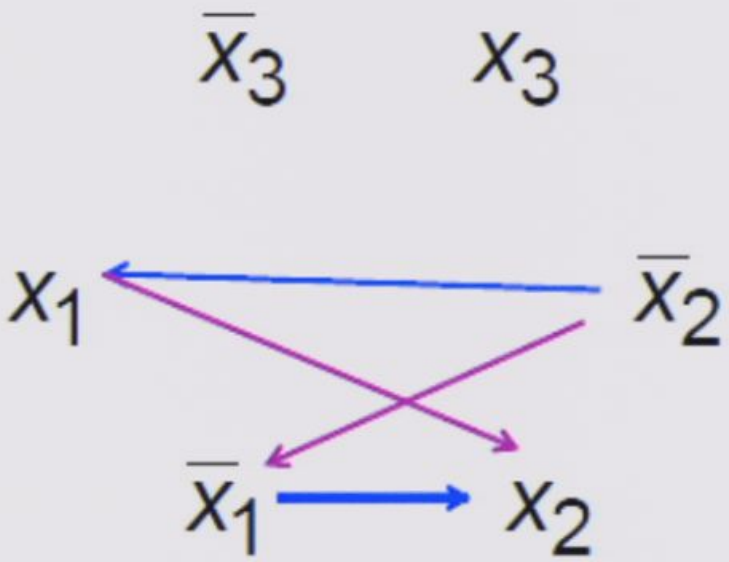


2-SAT is in P

Linear time algorithm for 2-SAT: turn implications into links of a graph. Instance is unsatisfiable if there is path from x_i to (not x_i) and from (not x_i) to x_i for some i, and satisfiable otherwise

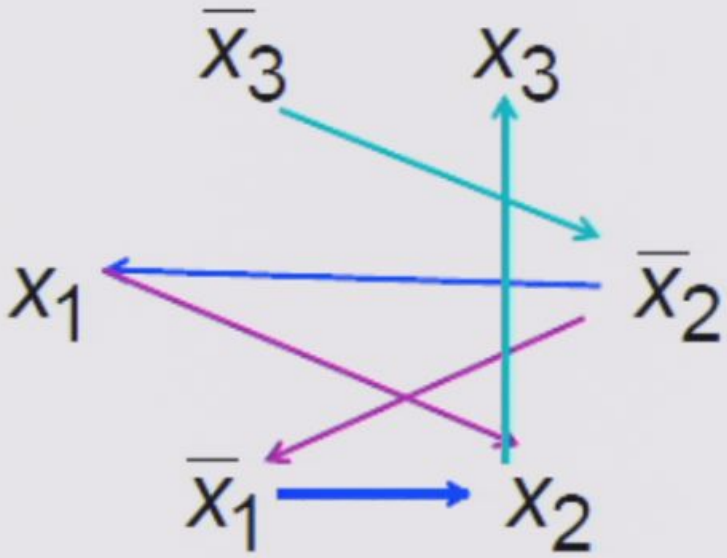
$(x_1 \text{ OR } x_2)$

AND $(\bar{x}_1 \text{ OR } x_2)$



2-SAT is in P

Linear time algorithm for 2-SAT: turn implications into links of a graph. Instance is unsatisfiable if there is path from x_i to (not x_i) and from (not x_i) to x_i for some i, and satisfiable otherwise



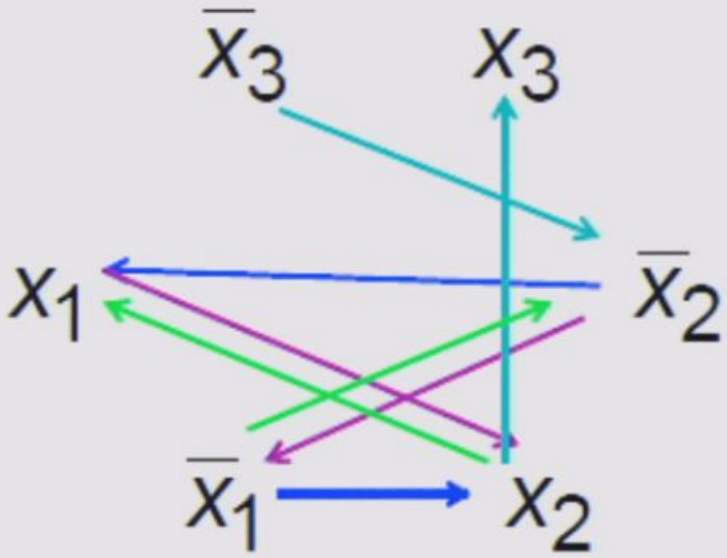
$(x_1 \text{ OR } x_2)$

AND $(\overline{x}_1 \text{ OR } x_2)$

AND $(\overline{x}_2 \text{ OR } x_3)$

2-SAT is in P

Linear time algorithm for 2-SAT: turn implications into links of a graph. Instance is unsatisfiable if there is path from x_i to (not x_i) and from (not x_i) to x_i for some i, and satisfiable otherwise



$(x_1 \text{ OR } x_2)$

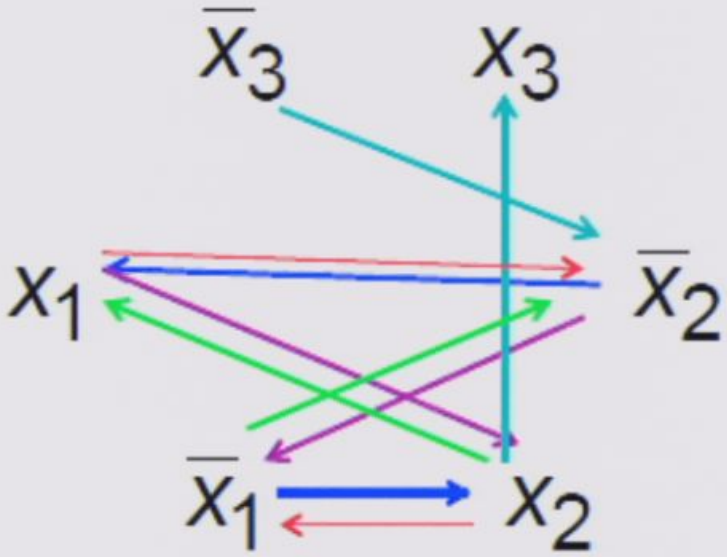
AND $(\bar{x}_1 \text{ OR } x_2)$

AND $(\bar{x}_2 \text{ OR } x_3)$

AND $(x_1 \text{ OR } \bar{x}_2)$

2-SAT is in P

Linear time algorithm for 2-SAT: turn implications into links of a graph. Instance is unsatisfiable if there is path from x_i to (not x_i) and from (not x_i) to x_i for some i, and satisfiable otherwise



$(x_1 \text{ OR } x_2)$

AND $(\bar{x}_1 \text{ OR } x_2)$

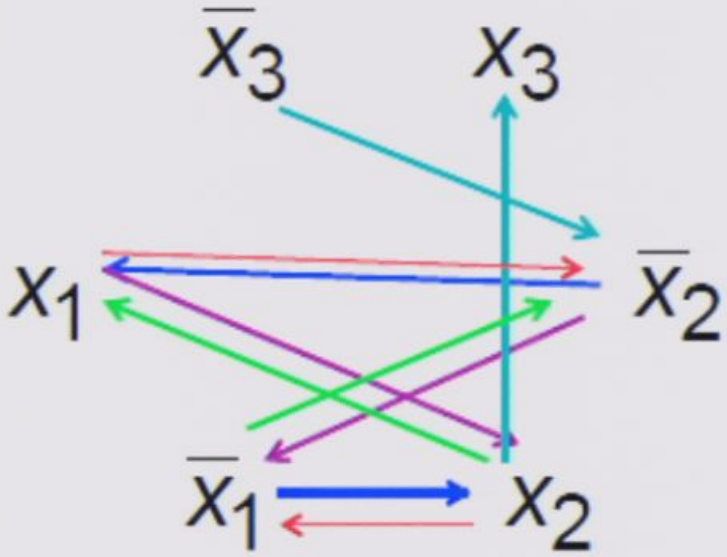
AND $(\bar{x}_2 \text{ OR } x_3)$

AND $(x_1 \text{ OR } \bar{x}_2)$

AND $(\bar{x}_1 \text{ OR } \bar{x}_2)$

2-SAT is in P

Linear time algorithm for 2-SAT: turn implications into links of a graph. Instance is unsatisfiable if there is path from x_i to (not x_i) and from (not x_i) to x_i for some i, and satisfiable otherwise



$(x_1 \text{ OR } x_2)$

AND $(\bar{x}_1 \text{ OR } x_2)$

AND $(\bar{x}_2 \text{ OR } x_3)$

AND $(x_1 \text{ OR } \bar{x}_2)$

AND $(\bar{x}_1 \text{ OR } \bar{x}_2)$

Summary

Key question: How do computational resources required to solve a problem scale with the size of the problem specification?

Complexity classes: sets of problems that require resources that are bounded in a specified fashion as the size of the problem is increased.

e.g., P – problem can be solved with polynomially bounded resources

NP – problem solution can be verified using polynomially bounded resources

There are many other complexity classes, e.g.,

PSPACE – problem can be solved in polynomially bounded space (but perhaps exponential time)

The complexity class PPAD

(PPAD stands for "polynomial parity argument for directed graphs")

Problems in PPAD are in NP, and are guaranteed to have a solution. But no efficient algorithm for finding a solution is known.

Original example (Papadimitriou 1994): given a directed graph with a vertex has an unequal number of ingoing and outgoing links, find a second "unbalanced" vertex. It is easy to prove that such a vertex must exist, but no efficient algorithm for finding it is known.


Microsoft
Windows xp
Professional

The complexity class PPAD

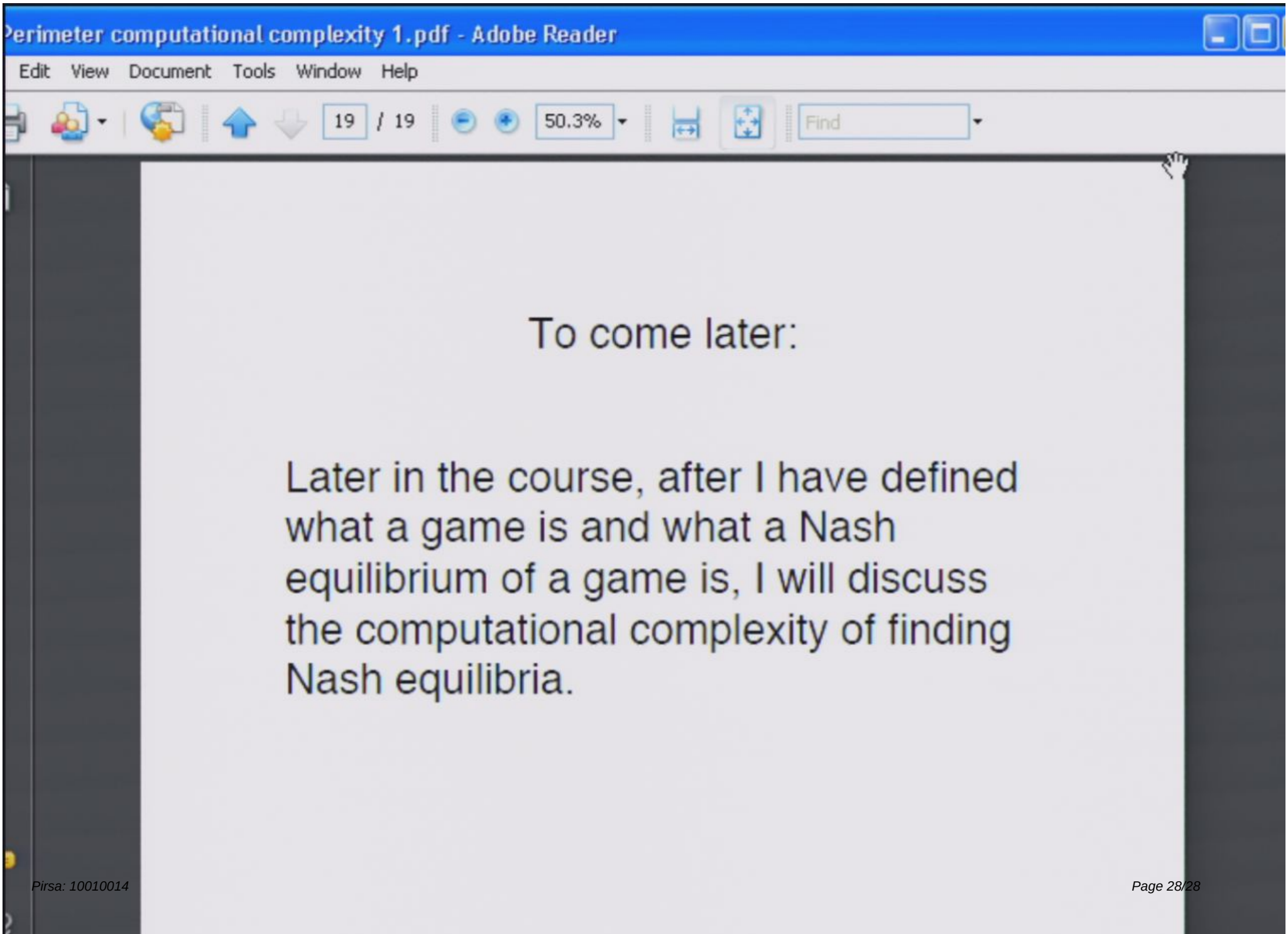
(PPAD stands for "polynomial parity argument for directed graphs")

Problems in PPAD are in NP, and are guaranteed to have a solution. But no efficient algorithm for finding a solution is known.

Original example (Papadimitriou 1994): given a directed graph with a vertex has an unequal number of ingoing and outgoing links, find a second "unbalanced" vertex. It is easy to prove that such a vertex must exist, but no efficient algorithm for finding it is known.

To come later:

Later in the course, after I have defined what a game is and what a Nash equilibrium of a game is, I will discuss the computational complexity of finding Nash equilibria.


Perimeter computational complexity 1.pdf - Adobe Reader
Edit View Document Tools Window Help
19 / 19
50.3%
Find
To come later:
Later in the course, after I have defined what a game is and what a Nash equilibrium of a game is, I will discuss the computational complexity of finding Nash equilibria.