

P vs. NP

Moshe Y. Vardi

Rice University

An Outstanding Open Problem

Does $P = NP$?

- *The* major open problem in computer science
- A major open problem in mathematics
 - A Clay Institute Millennium Problem
 - Million dollar prize!
- On August 6, 2010, Vinay Deolalikar announced a proof (100-page manuscript) that $P \neq NP$.

What is this about? It is about computational complexity – how hard it is to solve computational problems.

Rally To Restore Sanity, Washington, DC, October 2010



Computational Problems

Example: *Graph* – $G = (V, E)$

- V – set of nodes
- E – set of edges

Two notions:

- **Hamiltonian Cycle:** a cycle that visits every *node* exactly once.
- **Eulerian Cycle:** a cycle that visits every *edge* exactly once.

Question: How hard it is to find a Hamiltonian cycle? Eulerian cycle?

Figure 1: The Bridges of Königsburg

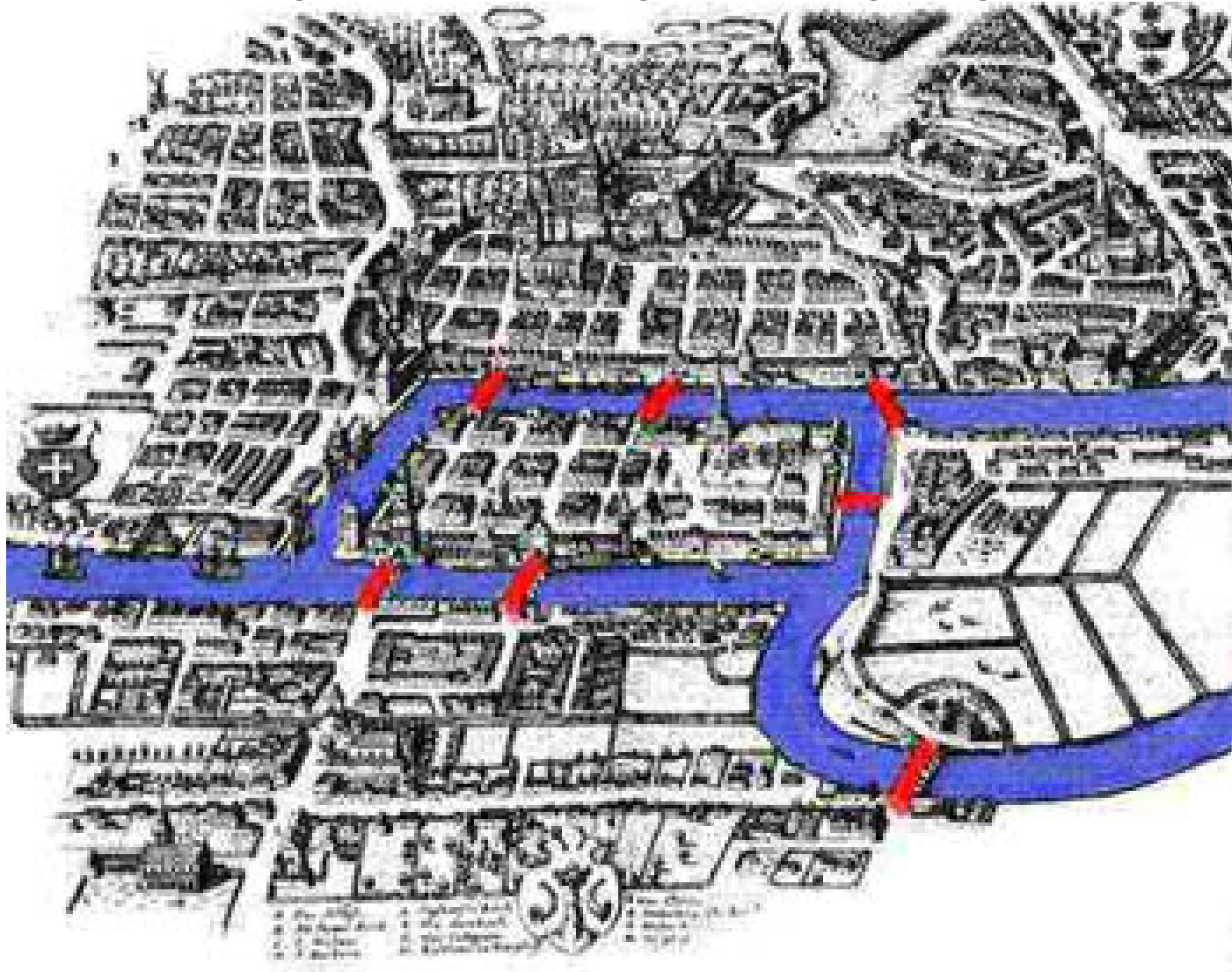


Figure 2: The Graph of The Bridges of Königsburg

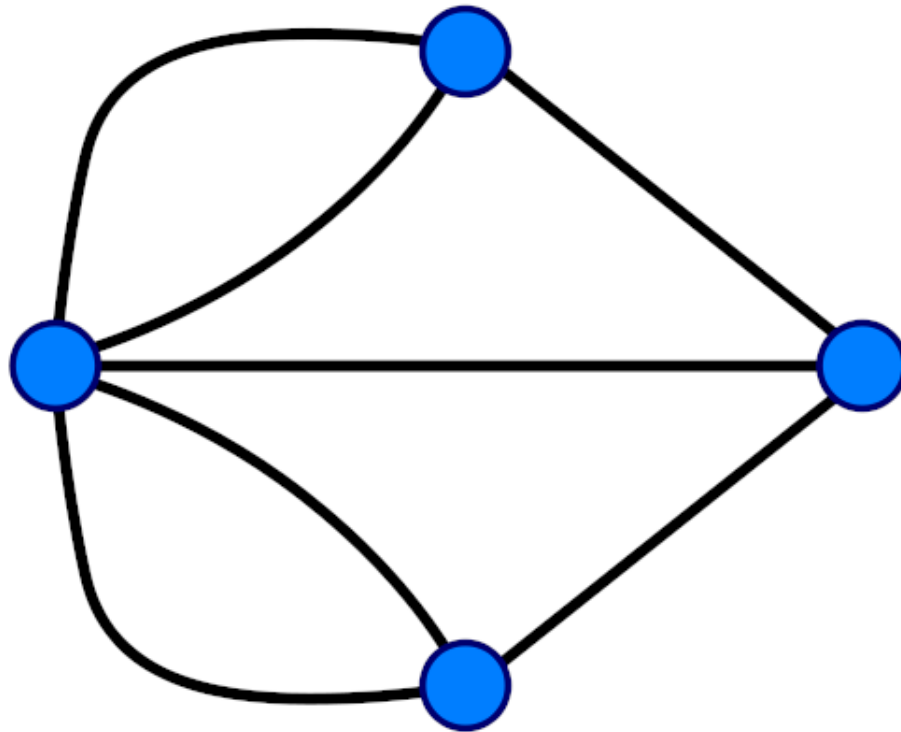
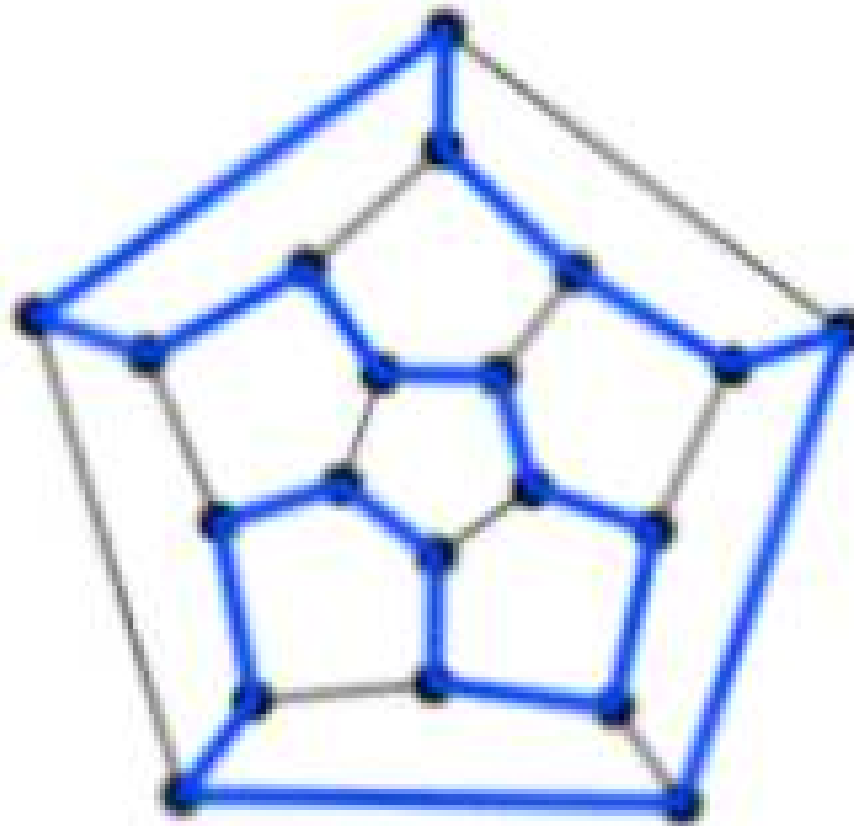


Figure 3: Hamiltonian Cycle



Computational Complexity

Measuring complexity: How many (Turing machine) operations does it take to solve a problem of size n ?

- *Size* of (V, E) : number of nodes plus number of edges.

Complexity Class P : problems that can be solved in *polynomial time*
– n^c for a *fixed* c

Examples:

- Is a number even?
- Is a number square?
- Does a graph have an Eulerian cycle?

What about the Hamiltonian Cycle Problem?

Hamiltonian Cycle

- **Naive Algorithm:** Exhaustive search – run time is $n!$ operations
- **“Smart” Algorithm:** Dynamic programming – run time is 2^n operations

Note: The universe is much younger than 2^{200} Planck time units!

Fundamental Question: Can we do better?

- Is HamiltonianCycle in P ?

Checking Is Easy!

Observation: Checking if a *given* cycle is a Hamiltonian cycle of a graph $G = (V, E)$ is *easy*!

Complexity Class NP : problems where solutions can be *checked* in polynomial time.

Examples:

- HamiltonianCycle
- Factoring numbers

Significance: Tens of thousands of optimization problems are in $NP!!!$

- CAD, flight scheduling, chip layout, protein folding, ...

P vs. NP

- P : efficient *discovery* of solutions
- NP : efficient *checking* of solutions

The Big Question: Is $P = NP$ or $P \neq NP$?

- Is *checking* really easier than *discovering*?

Intuitive Answer: Of course, *checking* is easier than *discovering*, so $P \neq NP$!!!

- **Metaphor:** finding a needle in a haystack
- **Metaphor:** Sudoku
- **Metaphor:** mathematical proofs

Alas: We do not know how to prove that $P \neq NP$.

$$P \neq NP$$

Consequences:

- Cannot solve efficiently numerous important problems
- RSA encryption may be safe.

Question: Why is it so important to *prove* $P \neq NP$, if that is what is commonly believed?

Answer:

- If we cannot prove it, we do not really understand it.
- May be $P = NP$ and the “enemy” proved it and broke RSA!

$$P = NP$$

S. Aaronson, MIT: “If $P = NP$, then the world would be a profoundly different place than we usually assume it to be. There would be no special value in ‘creative leaps,’ no fundamental gap between solving a problem and recognizing the solution once its found. Everyone who could appreciate a symphony would be Mozart; everyone who could follow a step-by-step argument would be Gauss.”

Consequences:

- Can solve efficiently numerous important problems.
- RSA encryption is not safe.

Question: Is it really possible that $P = NP$?

Answer: Yes! It'd require discovering a very clever algorithm, but it took 40 years to prove that LinearProgramming is in P .

Sharpening The Problem

NP -Complete Problems: hardest problems is NP

- HamiltonianCycle is NP -complete!

Corollary: $P = NP$ if and only if HamiltonianCycle is in P

There are *thousands* of NP -complete problems. To resolve the $P = NP$ question, it'd suffice to prove that *one* of them is or is not in P .

History

- 1950-60s: Futile effort to show hardness of search problems.
- Stephen Cook, 1971: Boolean Satisfiability is NP-complete.
- Richard Karp, 1972: 20 additional NP-complete problems— 0-1 Integer Programming, Clique, Set Packing, Vertex Cover, Set Covering, Hamiltonian Cycle, Graph Coloring, Exact Cover, Hitting Set, Steiner Tree, Knapsack, Job Scheduling, ...
 - *All* NP-complete problems are polynomially equivalent!
- Leonid Levin, 1973 (independently): Six NP-complete problems
- M. Garey and D. Johnson, 1979: “Computers and Intractability: A Guide to NP-Completeness” - hundreds of NP-complete problems.
- Clay Institute, 2000: \$1M Award!

Terminology

Terminological Chaos: The standard terminology did not converge until 1974.

Knuth, 1974, "A terminological Proposal"

- *Competing terms:* arduous, bad, costly, difficult, exorbitant, exparent, formidable, heavy, Herculean, impractical, interminable, intractable, obdurate, perarduous, polychronious, prodigious, Sisyphean, tricky.
- *Winning terms:* NP-hard and NP-complete.

Logic and Complexity

Richard Lipton, Blog, Aug. 8, 2010:

“At the highest level he is using the characterization of polynomial time via finite-model theory. His proof uses the beautiful result of Moshe Vardi (1982) and Neil Immerman (1986).”

Theorem: On ordered structures, a relation is defined by a first-order formula plus the Least Fixed Point (LFP) operator if and only if it is computable in polynomial time.

Paper: “The complexity of relational query languages”, 1982 > 1100 citations.

Terminology:

- *Relation*: set of tuples of elements, e.g., $<$ is set of pairs
- *Model Theory*: logical theory of mathematical structures – branch of mathematical logic
- *Finite-Model Theory*: logical theory of *finite* mathematical structures – between mathematical logic and computer science

The Language of Mathematics

G. Frege, *Begriffsschrift*, 1879: a universal mathematical language – *first-order logic*

- Objects, e.g., numbers
- Predicates (relationships), e.g., $2 < 3$
- Operations (functions), e.g., $2 + 3$
- Boolean operations: “and” (\wedge), “or” (\vee), “not” (\neg), “implies” (\rightarrow)
- Quantifiers: “for all” ($\forall x$), “there exists” ($\exists x$)

Back to Aristotle:

- “All men are mortal”
- “For all x , if x is a man, then x is mortal”
- $(\forall x)(Man(x) \rightarrow Mortal(x))$

First-Order Logic on Graphs

Syntax:

- Variables: x, y, z, \dots (range over nodes)
- Atomic formulas: $E(x, y), x = y$
- Formulas: Atomic Formulas + Boolean Connectives + First-Order Quantifiers

Examples:

- φ_1 : “node x has at least two distinct neighbors”

$$(\exists y)(\exists z)(\neg(y = z) \wedge E(x, y) \wedge E(x, z))$$

- φ_2 : “nodes x and y are connected by a path of length two”:

$$(\exists z)(E(x, z) \wedge E(z, y))$$

Formulas as Queries:

- φ_1 “computes” the set of nodes with at least two distinct neighbors.
- φ_2 “computes” the set of pairs of nodes connected by a path of length two.

Logic and Complexity

Theorem: [Immerman-V.]: Polynomial time computability is equivalent to computability by iterating positive first-order queries.

Significance:

- Machine-free characterization of P
 - **Note:** No Turing machines, no polynomial, no time!
- Normal form for P

Positivity

- *Positive*: φ_2 : “nodes x and y are connected by a path of length two”:

$$(\exists z)(E(x, z) \wedge E(z, y))$$

- *Non-Positive*: φ_3 : “nodes x and y are connected by an incomplete triangle”:

$$(\exists z)(E(x, y) \wedge E(x, z) \wedge \neg E(y, z))$$

Significance of Positivity: Iteration yields an increasing sequence of relations, guaranteeing convergence.

Example: 2-Colorability

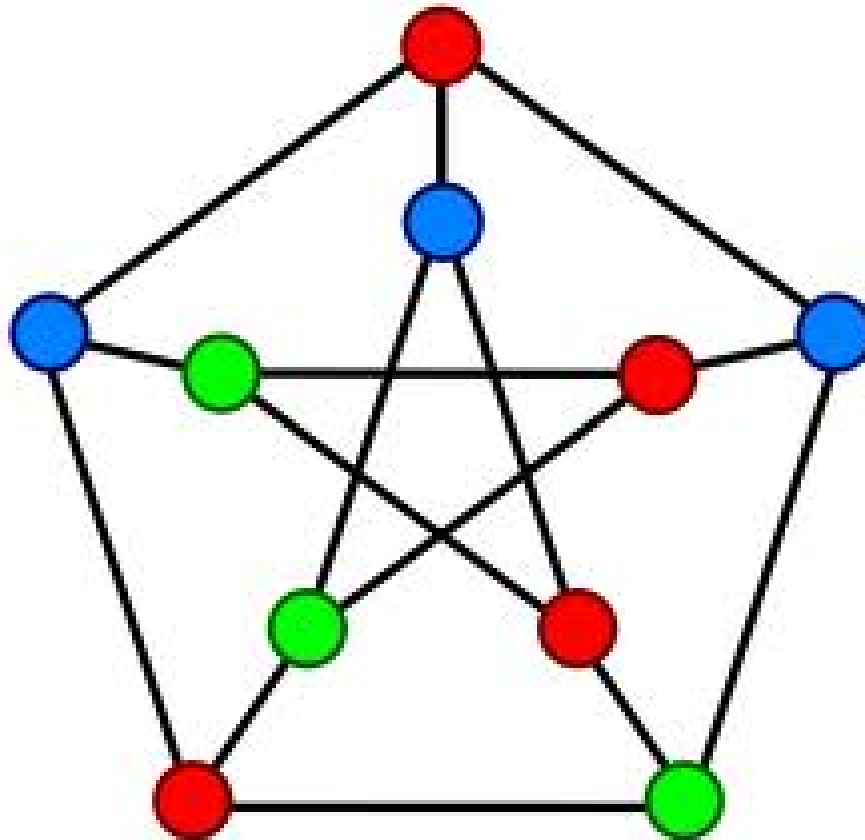
Graph Coloring:

- *Graph* – $G = (V, E)$
- *k-coloring*: $h : V \rightarrow \{1, \dots, k\}$
- *Nonmonochromaticity*: $h(u) \neq h(v)$ for all $(u, v) \in E$
- *k-Colorability*: Does G have k -coloring?

Complexity:

- 3-Colorability is NP-complete.
- 2-Colorability is in PTIME.

Figure 4: 3-Coloring



2-Colorability

Fact: A graph is 2-colorable iff it has no cycle of odd length.

Example: Logical characterization of non-2-colorability

$$O(X, Y) \leftarrow E(X, Y)$$

$$O(X, Y) \leftarrow O(X, Z), E(Z, W), E(W, Y)$$

$$\text{Not2Colorable} \leftarrow O(X, X)$$

Another Connection between Logic and Complexity

Boolean Satisfiability (SAT); Given a Boolean expression in the form of “and of ors”, is there a *satisfying* solution (an assignment of 0's and 1's to the variables that makes the expression equal 1)?

Example:

$$(\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_2 \vee \neg x_3 \vee x_4) \wedge (x_3 \vee x_1 \vee x_4)$$

Solution: $x_1 = 0, x_2 = 0, x_3 = 1, x_4 = 1$

Cook-Levin Theorem: SAT is NP-complete

A Physics Perspective

- *Literal*: Positive or negative variable – $x_1, \neg x_2$
- *Clause*: Disjunction (or) of literals – $(\neg x_1 \vee x_2 \vee x_3)$

Energy State:

- Satisfied clause: 0
- Unsatisfied clause: 1
- Total energy: sum of clausal energies = number of *unsatisfied* clauses

Physics Perspective: Does expression have a zero-energy state?

- Formula satisfied \Leftrightarrow zero-energy state

k -SAT

k -SAT:

- Each clause contains precisely k literals.
- 2-SAT is in P .

$$(\neg x_1 \vee x_2) \wedge (\neg x_2 \vee \neg x_3) \wedge (x_1 \vee x_4)$$

- k -SAT is NP -complete for $k > 2$.

Random k -SAT

Random k -SAT:

- Parameters:
 - number of variables – n ,
 - number of clauses – m
- m/n = Number of clauses divided by number of variables: **density** – fixed!
- Choose clauses at random, uniformly
- Limit: $n, m \rightarrow \infty$

Evolution of Random k -SAT

Intuition: Density analogous to temperature

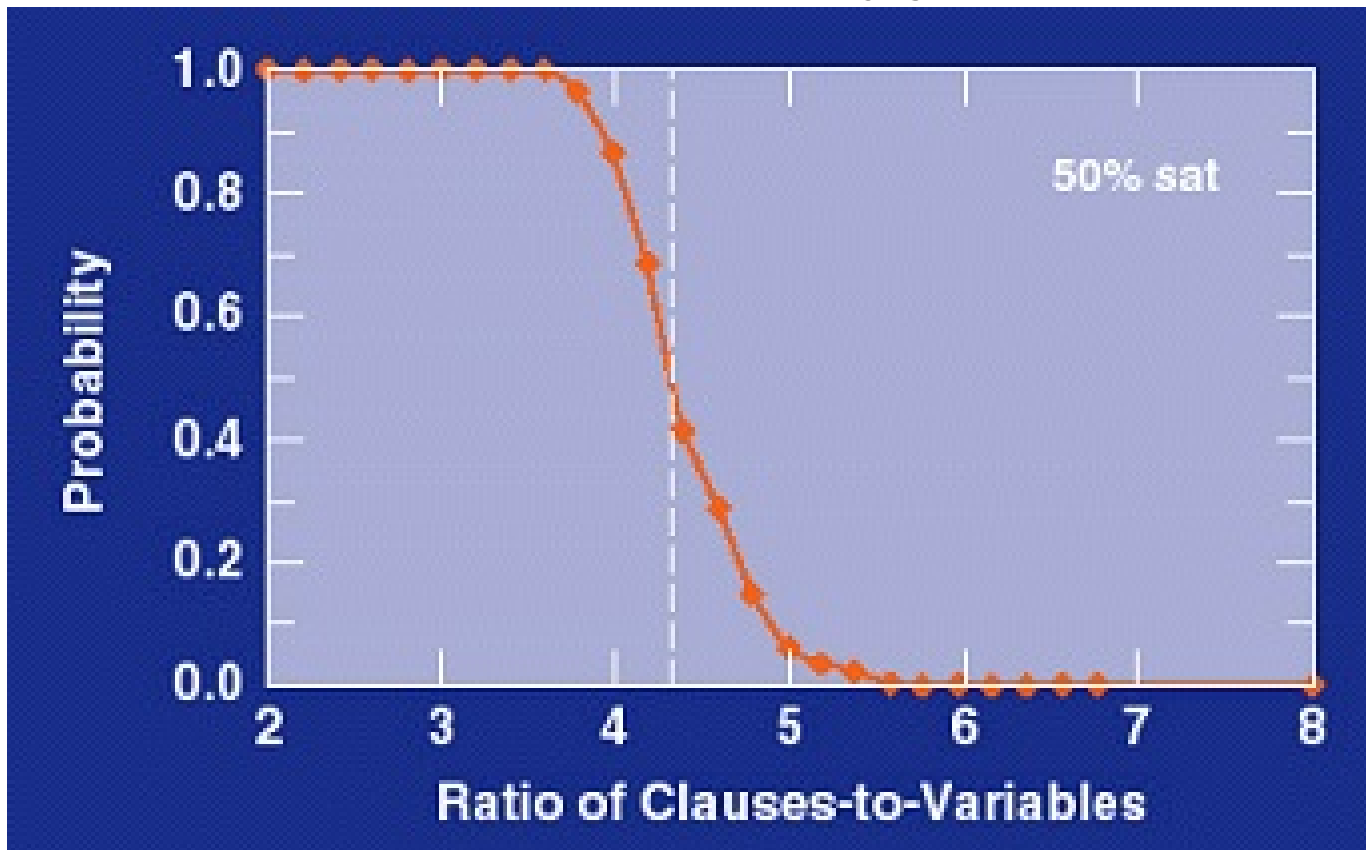
- *Low density:* low energy state – high probability of satisfiability – limit = 1
- *High density:* high energy state – low probability of satisfiability – limit = 0

Empirical Observation: Phase transition – limit probability drops from 1 to 0

- 2-SAT: phase transition at density 1 (also proved formally)
- 3-SAT: phase transition at density 4.26

1991-2010: Extensive research on statistical behavior of Random k -SAT

Phase Transition of 3-SAT



Essence of V.D.'s Proof

Crux: 9-SAT can not be in P !

- If 9-SAT is in P , then it can be expressed in $FO+LFP$, by the Immerman-V. Theorem.
- But, the $FO+LFP$ normal form is inconsistent with what is known about statistical behavior of random 9-SAT.

Reaction to Proof Announcement

A huge buzz!!!

Why?

- People announce solutions of the problem all the time.
- Every few months paper posted on arXiv.org.

But:

- V.D. is a Principal Research Scientist at HP.
- Stephen Cook (founding figure in complexity theory): “This appears to be a relatively serious claim”
- Nice connection of complexity, logic, and physics!
- Richard Lipton (senior complexity theorist and influential blogger): Blog item on August 8, 2010, slashdotted

Proof Checking at The Internet Age

“Ten Days of Fame”: Proof discredited in ten days!

- Aug. 6: Manuscript sent to 22 people and put on web page
- Aug. 7: First blog post [Greg Baker]
- Aug. 8: Second blog post [Richard Lipton], Slashdot
 - extensive commentary
- Aug. 9: Wikipedia article about V.D. (deleted later)
- Aug. 10: Wiki for technical discussion established
 - hundreds of edits
 - Fields medalists involved
- Aug. 15: CACM blogpost by Lipton
- Aug.16: New York Times article

The Flaw

A major problem: V.D.'s proof does not seem to distinguish between intractable and tractable cases of k -SAT.

Cause: Misuse of the Immerman-V. Theorem.

A Tractable Fragment of SAT

Affine Boolean Satisfiability (Affine SAT): Given a Boolean expression in the form of “and of xors”, is there a *satisfying* solution (an assignment of 0’s and 1’s to the variables that makes the expression equal 1)?

Example:

$$(\neg x_1 \oplus x_2 \oplus x_3) \wedge (\neg x_2 \oplus \neg x_3 \oplus x_4) \wedge (x_3 \oplus x_1 \oplus x_4)$$

In essence: Linear equations modulo 2

- Solve using Gaussian elimination

But: Random k -SAT and Random Affine k -SAT are quite similar statistically!

Revision at the Internet Age

- First draft, Aug. 6
- Second draft Aug. 9–11
- Third draft, Aug. 11–17
- All drafts removed after Aug 17

Consensus: The P vs. NP problem withstood another challenge and remained wide open!

- Wikipedia: “However, the general consensus amongst theoretical computer scientists is now that the attempted proof is not correct, nor even a significant advancement in our understanding of the problem.”

No Concession!

From V.D.'s website:

“The preliminary version was meant to solicit feedback from a few researchers as is customarily done. It illustrated the interplay of principles from various areas, which was the major effort in constructing the proof. I have fixed all the issues that were raised about the preliminary version in a revised manuscript; clarified some concepts; and obtained simpler proofs of several claims. Once I hear back from the journal as part of due process, I will put up the final version on this website.”

Reflection on P vs. NP

Old Cliché “What is the difference between theory and practice? In theory, they are not that different, but in practice, they are quite different.”

P vs. NP in practice:

- $P=NP$: Conceivably, NP-complete problems can be solved in polynomial time, but the polynomial is $(10n)^{1000}$ – *impractical!*
- $P \neq NP$: Conceivably, NP-complete problems can be solved by $n^{\log \log \log n}$ operations – *practical!*

Conclusion: No guarantee that solving P vs. NP would yield practical benefits.

Theory, Practice & Programming

- **Theory:** You know something, but it doesn't work.
- **Practice:** Something works, but you don't know why
- **Programming:** Combine theory and practice: Nothing works, and we don't know why!

Are NP-Complete Problems Really Hard?

- When I was a graduate student, SAT was a “scary” problem, not to be touched with a 10-foot pole.
- Indeed, there are SAT instances with a few hundred variables that cannot be solved by any extant SAT solver.
- But today’s SAT solvers, which enjoy wide industrial usage, routinely solve real-life SAT instances with over one million variables!

Conclusion We need a richer and broader complexity theory, a theory that would explain both the difficulty and the easiness of problems like SAT.