Intractability

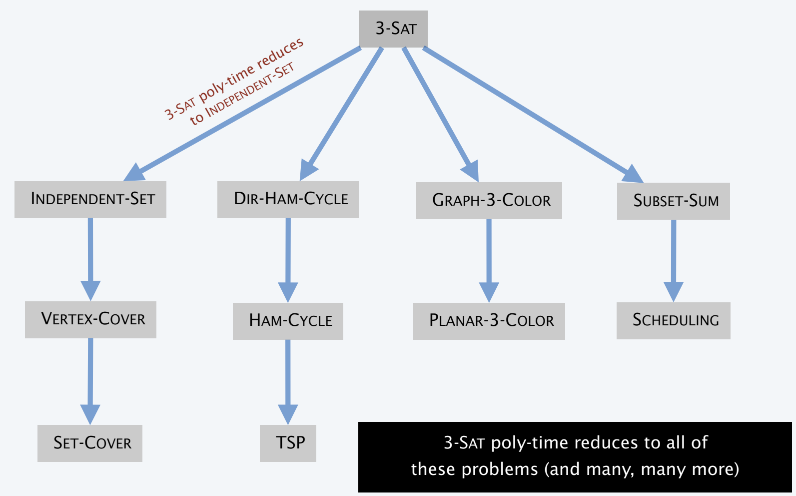
P,NP,NP-Complete

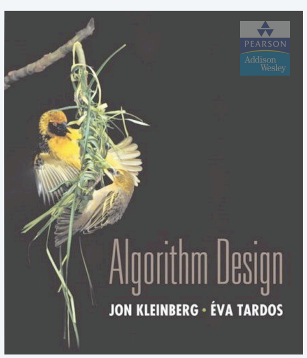
Tyler Moore

CS 2123, The University of Tulsa

Some slides created by or adapted from Dr. Kevin Wayne. For more information see `http://www.cs.princeton.edu/~wayne/kleinberg-tardos`. Some code reused from Python Algorithms by Magnus Lie Hetland.

Recap





SECTION 8.3

8. INTRACTABILITY II

- *P vs. NP*
- *NP-complete*
- *co-NP*
- *NP-hard*

Decision problems

Decision problem.

- Problem X is a set of strings.
- Instance s is one string.
- Algorithm A solves problem X: $A(s) = yes$ iff $s \in X$.

Def. Algorithm A runs in polynomial time if for every string s, $A(s)$ terminates in at most $p(|s|)$ "steps", where $p(\cdot)$ is some polynomial.

↑ length of s

Ex.

- Problem PRIMES = $\{ 2, 3, 5, 7, 11, 13, 17, 23, 29, 31, 37, \ldots. \}$.
- Instance $s = 592335744548702854681$.
- AKS algorithm PRIMES in $O(|s|^8)$ steps.

Definition of P

P. Decision problems for which there is a poly-time algorithm.

Problem	Description	Algorithm	yes	no
MULTIPLE	Is x a multiple of y ?	grade-school division	51, 17	51, 16
REL-PRIME	Are x and y relatively prime ?	Euclid (300 BCE)	34, 39	34, 51
PRIMES	Is x prime ?	AKS (2002)	53	51
EDIT-DISTANCE	Is the edit distance between x and y less than 5 ?	dynamic programming	niether neither	acgggt ttttta
L-SOLVE	Is there a vector x that satisfies $Ax = b$?	Gauss-Edmonds elimination	$\begin{bmatrix} 0 & 1 & 1 \\ 2 & 4 & -2 \\ 0 & 3 & 15 \end{bmatrix}, \begin{bmatrix} 4 \\ 2 \\ 36 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$
ST-CONN	Is there a path between s and t in a graph G ?	depth-first search (Theseus)		

NP

Certification algorithm intuition.
- Certifier views things from "managerial" viewpoint.
- Certifier doesn't determine whether $s \in X$ on its own; rather, it checks a proposed proof t that $s \in X$.

Def. Algorithm $C(s, t)$ is a certifier for problem X if for every string s, $s \in X$ iff there exists a string t such that $C(s, t) = yes$.

"certificate" or "witness"

Def. **NP** is the set of problems for which there exists a poly-time certifier.
- $C(s, t)$ is a poly-time algorithm.
- Certificate t is of polynomial size: $|t| \leq p(|s|)$ for some polynomial $p(\cdot)$

Remark. **NP** stands for nondeterministic polynomial time.

Certifiers and certificates: composite

COMPOSITES. Given an integer s, is s composite?

Certificate. A nontrivial factor t of s. Such a certificate exists iff s is composite. Moreover $|t| \leq |s|$.

Certifier. Check that $1 < t < s$ and that s is a multiple of t.

instance s	437669
certificate t	541 or 809

437,669 = 541 × 809

Conclusion. COMPOSITES ∈ **NP**.

Certifiers and certificates: 3-satisfiability

3-SAT. Given a CNF formula Φ, is there a satisfying assignment?

Certificate. An assignment of truth values to the n boolean variables.

Certifier. Check that each clause in Φ has at least one true literal.

instance s $\Phi = (\overline{x_1} \vee x_2 \vee x_3) \wedge (x_1 \vee \overline{x_2} \vee x_3) \wedge (\overline{x_1} \vee x_2 \vee x_4)$

certificate t $x_1 = true, \; x_2 = true, \; x_3 = false, \; x_4 = false$

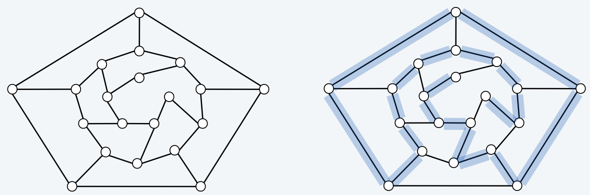
Conclusion. 3-SAT ∈ **NP**.

Certifiers and certificates: Hamilton path

HAM-PATH. Given an undirected graph $G = (V, E)$, does there exist a simple path P that visits every node?

Certificate. A permutation of the n nodes.

Certifier. Check that the permutation contains each node in V exactly once, and that there is an edge between each pair of adjacent nodes.



instance s — certificate t

Conclusion. HAM-PATH $\in$ **NP**.

Definition of NP

NP. Decision problems for which there is a poly-time certifier.

Problem	Description	Algorithm	yes	no
L-SOLVE	Is there a vector x that satisfies $Ax = b$?	Gauss-Edmonds elimination	$\begin{bmatrix} 0 & 1 & 1 \\ 2 & 4 & -2 \\ 0 & 3 & 15 \end{bmatrix}, \begin{bmatrix} 4 \\ 2 \\ 36 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$
COMPOSITES	Is x composite ?	AKS (2002)	51	53
FACTOR	Does x have a nontrivial factor less than y ?	?	(56159, 50)	(55687, 50)
SAT	Is there a truth assignment that satisfies the formula ?	?	$\neg x_1 \lor x_2$ $x_1 \lor x_2$	$\neg x_2$ $\neg x_1 \lor x_2$ $x_1 \lor x_2$
3-COLOR	Can the nodes of a graph G be colored with 3 colors?	?		
HAM-PATH	Is there a simple path between s and t that visits every node?	?		

Definition of NP

NP. Decision problems for which there is a poly-time certifier.

> *" NP captures vast domains of computational, scientific, and mathematical endeavors, and seems to roughly delimit what mathematicians and scientists have been aspiring to compute feasibly. "* — *Christos Papadimitriou*

> *" In an ideal world it would be renamed P vs VP. "* — *Clyde Kruskal*

P, NP, and EXP

P. Decision problems for which there is a poly-time algorithm.
NP. Decision problems for which there is a poly-time certifier.
EXP. Decision problems for which there is an exponential-time algorithm.

Claim. **P** $\subseteq$ **NP**.
Pf. Consider any problem $X \in$ **P**.
- By definition, there exists a poly-time algorithm $A(s)$ that solves X.
- Certificate $t = \varepsilon$, certifier $C(s, t) = A(s)$. ▪

Claim. **NP** $\subseteq$ **EXP**.
Pf. Consider any problem $X \in$ **NP**.
- By definition, there exists a poly-time certifier $C(s, t)$ for X.
- To solve input s, run $C(s, t)$ on all strings t with $|t| \leq p(|s|)$.
- Return *yes* if $C(s, t)$ returns *yes* for any of these potential certificates. ▪

Remark. Time-hierarchy theorem implies **P** $\subsetneq$ **EXP**.

The main question: P vs. NP

Q. How to solve an instance of 3-SAT with n variables?
A. Exhaustive search: try all 2^n truth assignments.

Q. Can we do anything substantially more clever?
Conjecture. No poly-time algorithm for 3-SAT.

"intractable"

The main question: P vs. NP

Does P = NP? [Cook 1971, Edmonds, Levin, Yablonski, Gödel]
Is the decision problem as easy as the certification problem?

If P ≠ NP   If P = NP

If yes. Efficient algorithms for 3-SAT, TSP, 3-COLOR, FACTOR, …
If no. No efficient algorithms possible for 3-SAT, TSP, 3-COLOR, …

Consensus opinion. Probably no.

Possible outcomes

P ≠ NP.

> *" I conjecture that there is no good algorithm for the traveling salesman problem. My reasons are the same as for any mathematical conjecture: (i) It is a legitimate mathematical possibility and (ii) I do not know."*
> — *Jack Edmonds 1966*

Possible outcomes

P ≠ NP.

> *" In my view, there is no way to even make intelligent guesses about the answer to any of these questions. If I had to bet now, I would bet that P is not equal to NP. I estimate the half-life of this problem at 25–50 more years, but I wouldn't bet on it being solved before 2100. "*
> — *Bob Tarjan*

> *" We seem to be missing even the most basic understanding of the nature of its difficulty…. All approaches tried so far probably (in some cases, provably) have failed. In this sense P =NP is different from many other major mathematical problems on which a gradual progress was being constantly done (sometimes for centuries) whereupon they yielded, either completely or partially. "*
> — *Alexander Razborov*

Possible outcomes

P = NP.

“ P = NP. In my opinion this shouldn't really be a hard problem; it's just that we came late to this theory, and haven't yet developed any techniques for proving computations to be hard. Eventually, it will just be a footnote in the books. ” — *John Conway*

Other possible outcomes

P = **NP**, but only $\Omega(n^{100})$ algorithm for 3-SAT.

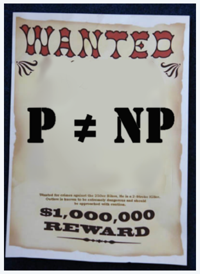
P ≠ **NP**, but with $O(n^{\log^* n})$ algorithm for 3-SAT.

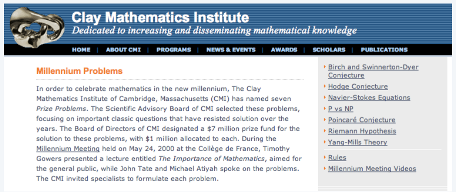
P = **NP** is independent (of ZFC axiomatic set theory).

“ It will be solved by either 2048 or 4096. I am currently somewhat pessimistic. The outcome will be the truly worst case scenario: namely that someone will prove “P = NP because there are only finitely many obstructions to the opposite hypothesis”; hence there will exists a polynomial time solution to SAT but we will never know its complexity! ” — *Donald Knuth*

Millennium prize

Millennium prize. $1 million for resolution of **P** = **NP** problem.

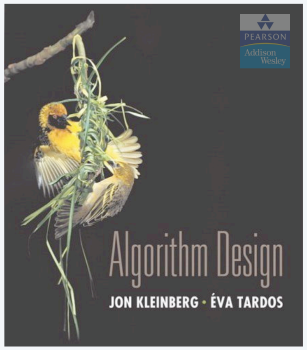




Looking for a job?

Some writers for the Simpsons and Futurama.

- J. Steward Burns. *M.S. in mathematics (Berkeley '93).*
- David X. Cohen. *M.S. in computer science (Berkeley '92).*
- Al Jean. *B.S. in mathematics. (Harvard '81).*
- Ken Keeler. *Ph.D. in applied mathematics (Harvard '90).*
- Jeff Westbrook. *Ph.D. in computer science (Princeton '89).*



SECTION 8.4

8. INTRACTABILITY II

- *P vs. NP*
- *NP-complete*
- *co-NP*
- *NP-hard*

Polynomial transformation

Def. Problem X polynomial (Cook) reduces to problem Y if arbitrary instances of problem X can be solved using:

- Polynomial number of standard computational steps, plus
- Polynomial number of calls to oracle that solves problem Y.

Def. Problem X polynomial (Karp) transforms to problem Y if given any input x to X, we can construct an input y such that x is a *yes* instance of X iff y is a *yes* instance of Y.

we require |y| to be of size polynomial in |x|

Note. Polynomial transformation is polynomial reduction with just one call to oracle for Y, exactly at the end of the algorithm for X. Almost all previous reductions were of this form.

Open question. Are these two concepts the same with respect to **NP**?

we abuse notation $\leq_p$ and blur distinction

NP-complete

NP-complete. A problem $Y \in$ **NP** with the property that for every problem $X \in$ **NP**, $X \leq_p Y$.

Theorem. Suppose $Y \in$ **NP**-complete. Then $Y \in$ **P** iff **P** = **NP**.

Pf. $\Leftarrow$ If **P** = **NP**, then $Y \in$ **P** because $Y \in$ **NP**.

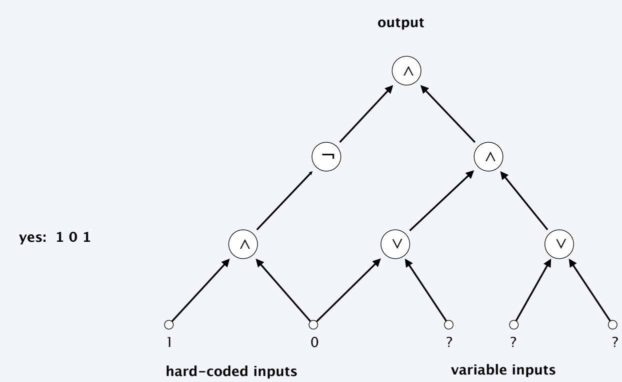
Pf. $\Rightarrow$ Suppose $Y \in$ **P**.

- Consider any problem $X \in$ **NP**. Since $X \leq_p Y$, we have $X \in$ **P**.
- This implies **NP** $\subseteq$ **P**.
- We already know **P** $\subseteq$ **NP**. Thus **P** = **NP**. ▪

Fundamental question. Do there exist "natural" **NP**-complete problems?

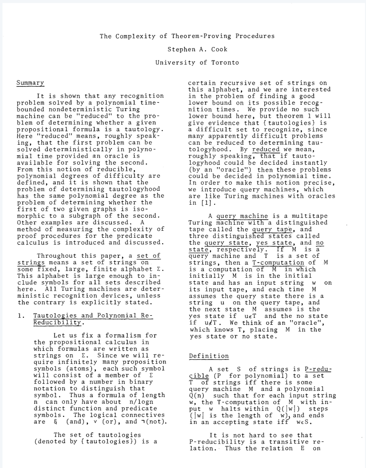
Circuit satisfiability

CIRCUIT-SAT. Given a combinational circuit built from AND, OR, and NOT gates, is there a way to set the circuit inputs so that the output is 1?



The "first" NP-complete problem

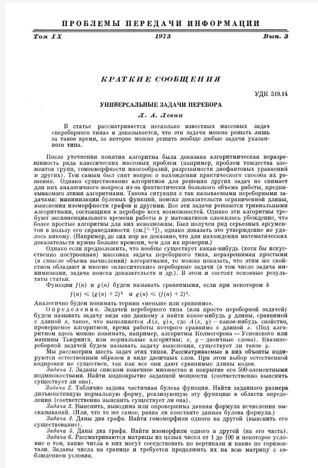
Theorem. CIRCUIT-SAT $\in$ **NP**-complete. [Cook 1971, Levin 1973]



The Complexity of Theorem-Proving Procedures

Stephen A. Cook

University of Toronto



ПРОБЛЕМЫ ПЕРЕДАЧИ ИНФОРМАЦИИ

Том IX 1973 Вып. 3

КРАТКИЕ СООБЩЕНИЯ

УДК 519.14

УНИВЕРСАЛЬНЫЕ ЗАДАЧИ ПЕРЕБОРА

Л. А. Левин

The "first" NP-complete problem

Theorem. CIRCUIT-SAT $\in$ **NP**-complete.

Pf sketch.

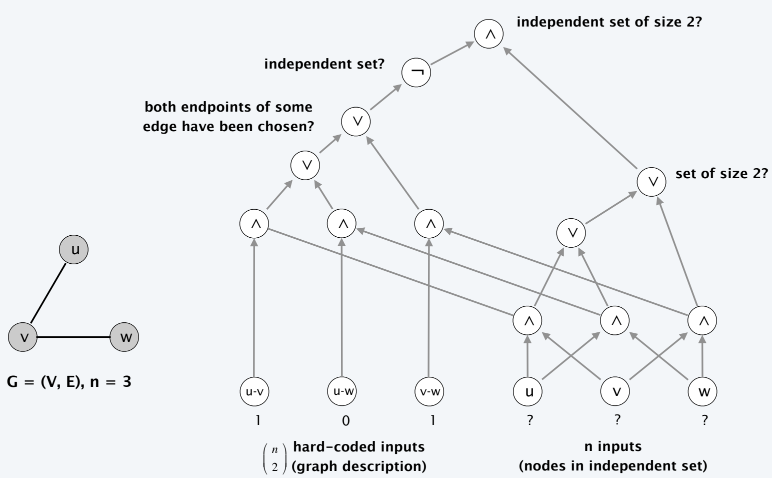
- Clearly, CIRCUIT-SAT $\in$ **NP**.
- Any algorithm that takes a fixed number of bits n as input and produces a *yes* or *no* answer can be represented by such a circuit.
- Moreover, if algorithm takes poly-time, then circuit is of poly-size.

sketchy part of proof; fixing the number of bits is important, and reflects basic distinction between algorithms and circuits

- Consider any problem $X \in$ **NP**. It has a poly-time certifier $C(s, t)$: $s \in X$ iff there exists a certificate t of length $p(|s|)$ such that $C(s, t) = yes$.
- View $C(s, t)$ as an algorithm with $|s| + p(|s|)$ input bits and convert it into a poly-size circuit K.
  - first $|s|$ bits are hard-coded with s
  - remaining $p(|s|)$ bits represent (unknown) bits of t
- Circuit K is satisfiable iff $C(s, t) = yes$.

Example

Ex. Construction below creates a circuit K whose inputs can be set so that it outputs 1 iff graph G has an independent set of size 2.



Establishing NP-completeness

Remark. Once we establish first "natural" **NP**-complete problem, others fall like dominoes.

Recipe. To prove that $Y \in$ **NP**-complete:

- Step 1. Show that $Y \in$ **NP**.
- Step 2. Choose an **NP**-complete problem X.
- Step 3. Prove that $X \leq_p Y$.

Theorem. If $X \in$ **NP**-complete, $Y \in$ **NP**, and $X \leq_p Y$, then $Y \in$ **NP**-complete.

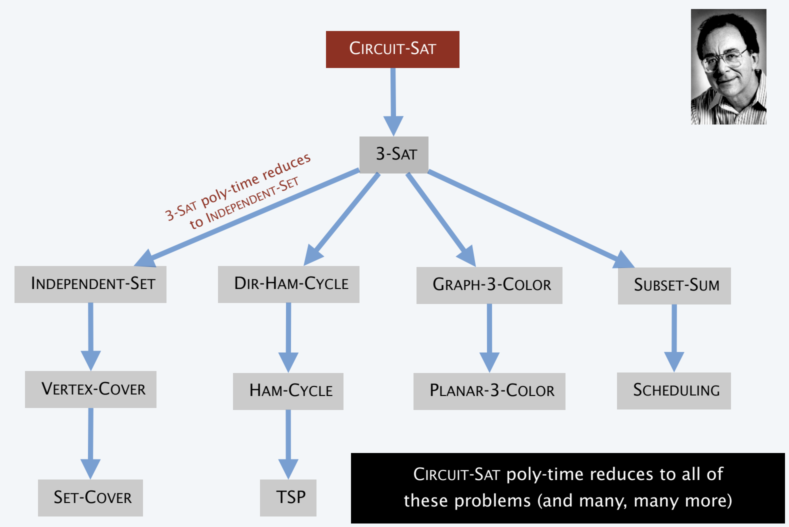
Pf. Consider any problem $W \in$ **NP**. Then, both $W \leq_p X$ and $X \leq_p Y$.

- By transitivity, $W \leq_p Y$.
- Hence $Y \in$ **NP**-complete. ▪

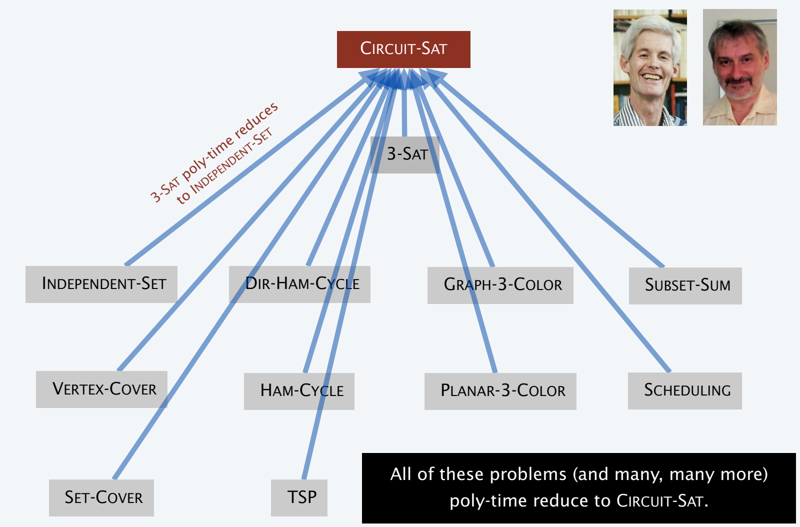
by definition of NP-complete

by assumption

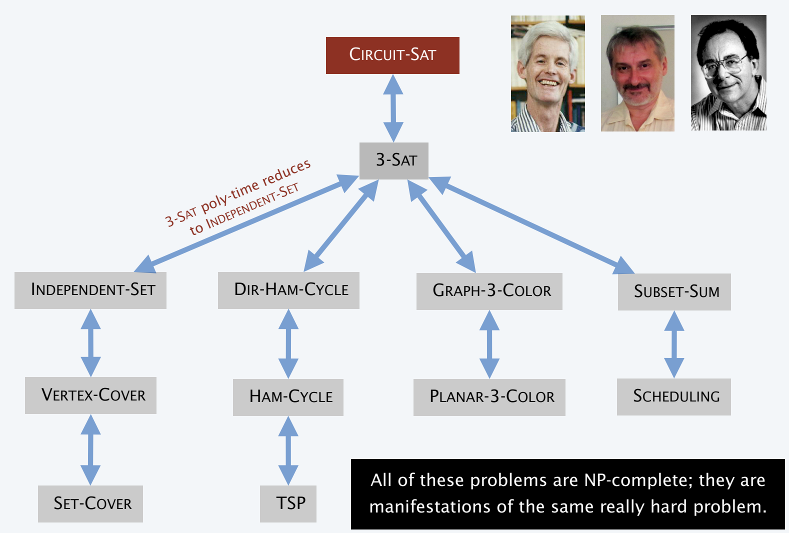
Implications of Karp



Implications of Cook-Levin



Implications of Karp + Cook-Levin



Some NP-complete problems

Basic genres of NP-complete problems and paradigmatic examples.

- Packing + covering problems: Set-Cover, Vertex-Cover, Independent-Set.
- Constraint satisfaction problems: Circuit-Sat, Sat, 3-Sat.
- Sequencing problems: Ham-Cycle, TSP.
- Partitioning problems: 3D-Matching, 3-Color.
- Numerical problems: Subset-Sum, Partition.

Practice. Most **NP** problems are known to be either in **P** or **NP**-complete.

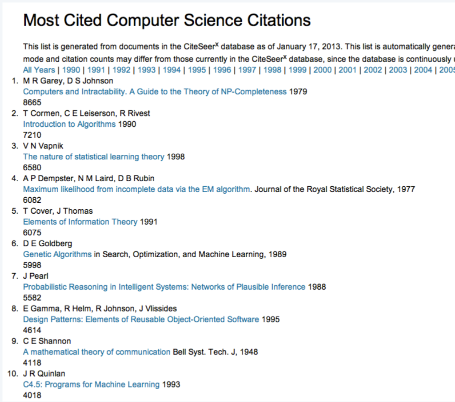
Notable exceptions. Factor, Graph-Isomorphism, Nash-Equilibrium.

Theory. [Ladner 1975] Unless **P** = **NP**, there exist problems in **NP** that are neither in **P** nor **NP**-complete.

More hard computational problems

Garey and Johnson. Computers and Intractability.

- Appendix includes over 300 **NP**-complete problems.
- Most cited reference in computer science literature.

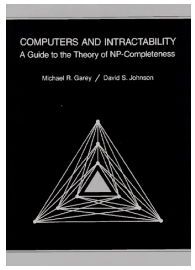


Most Cited Computer Science Citations

This list is generated from documents in the CiteSeer^X database as of January 17, 2013. This list is automatically generated and may contain errors. The list is generated in batch mode and citation counts may differ from those currently in the CiteSeer^X database, since the database is continuously updated.

All Years | 1990 | 1991 | 1992 | 1993 | 1994 | 1995 | 1996 | 1997 | 1998 | 1999 | 2000 | 2001 | 2002 | 2003 | 2004 | 2005 | 2006 | 2007 | 2008 | 2009 | 2010 | 2011 | 2012 | 2013

1. M R Garey, D S Johnson
   Computers and Intractability. A Guide to the Theory of NP-Completeness 1979
   8665
2. T Cormen, C E Leiserson, R Rivest
   Introduction to Algorithms 1990
   7210
3. V N Vapnik
   The nature of statistical learning theory 1998
   6580
4. A P Dempster, N M Laird, D B Rubin
   Maximum likelihood from incomplete data via the EM algorithm. Journal of the Royal Statistical Society, 1977
   6082
5. T Cover, J Thomas
   Elements of Information Theory 1991
   6075
6. D E Goldberg
   Genetic Algorithms in Search, Optimization, and Machine Learning, 1989
   5998
7. J Pearl
   Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference 1988
   5582
8. E Gamma, R Helm, R Johnson, J Vlissides
   Design Patterns: Elements of Reusable Object-Oriented Software 1995
   4614
9. C E Shannon
   A mathematical theory of communication Bell Syst. Tech. J, 1948
   4118
10. J R Quinlan
    C4.5: Programs for Machine Learning 1993
    4018

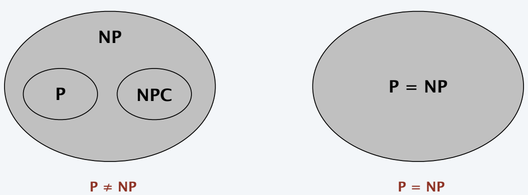


More hard computational problems

Aerospace engineering. Optimal mesh partitioning for finite elements.
Biology. Phylogeny reconstruction.
Chemical engineering. Heat exchanger network synthesis.
Chemistry. Protein folding.
Civil engineering. Equilibrium of urban traffic flow.
Economics. Computation of arbitrage in financial markets with friction.
Electrical engineering. VLSI layout.
Environmental engineering. Optimal placement of contaminant sensors.
Financial engineering. Minimum risk portfolio of given return.
Game theory. Nash equilibrium that maximizes social welfare.
Mathematics. Given integer $a_1, \ldots, a_n$, compute $\int_0^{2\pi} \cos(a_1\theta) \times \cos(a_2\theta) \times \cdots \times \cos(a_n\theta)\, d\theta$
Mechanical engineering. Structure of turbulence in sheared flows.
Medicine. Reconstructing 3d shape from biplane angiocardiogram.
Operations research. Traveling salesperson problem.
Physics. Partition function of 3d Ising model.
Politics. Shapley-Shubik voting power.
Recreation. Versions of Sudoko, Checkers, Minesweeper, Tetris.
Statistics. Optimal experimental design.

P vs. NP revisited

Overwhelming consensus (still). P ≠ NP.



Why we believe **P** ≠ **NP**.

> *" We admire Wiles' proof of Fermat's last theorem, the scientific theories of Newton, Einstein, Darwin, Watson and Crick, the design of the Golden Gate bridge and the Pyramids, precisely because they seem to require a leap which cannot be made by everyone, let alone a by simple mechanical device. "* — *Avi Wigderson*

You NP-complete me