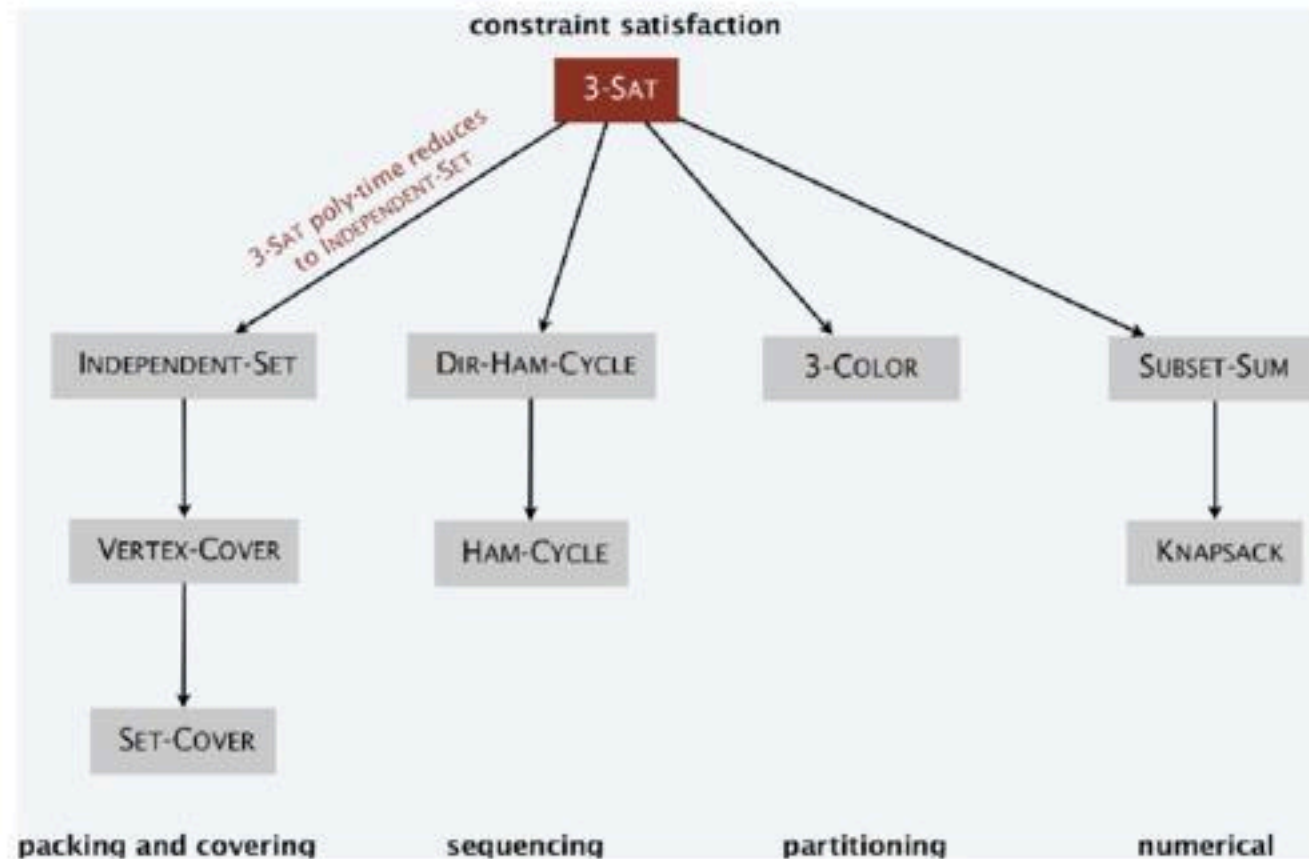


8. Intractability II

WU Xiaokun 吴晓堃

xkun.wu [at] gmail

Recap: Poly-time reductions



\mathcal{P} vs. \mathcal{NP}

\mathcal{P}

Decision problem.

- Problem X is a set of strings.
- Instance s is one string.
- Algorithm A solves problem X :

$$A(s) = \begin{cases} \text{yes} & \text{if } s \in X \\ \text{no} & \text{if } s \notin X \end{cases}$$

\mathcal{P}

Decision problem.

- Problem X is a set of strings.
- Instance s is one string.
- Algorithm A solves problem X :

$$A(s) = \begin{cases} \text{yes} & \text{if } s \in X \\ \text{no} & \text{if } s \notin X \end{cases}$$

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

Def. \mathcal{P} = set of decision problems for which there exists a poly-time algorithm.

\mathcal{P}

Decision problem.

- Problem X is a set of strings.
- Instance s is one string.
- Algorithm A solves problem X :

$$A(s) = \begin{cases} \text{yes} & \text{if } s \in X \\ \text{no} & \text{if } s \notin X \end{cases}$$

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

Def. \mathcal{P} = set of decision problems for which there exists a poly-time algorithm.


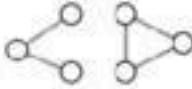
Ex. PRIMES

- problem: $\{2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, \dots\}$

- instance s : 592335744548702854681
- algorithm: Agrawal-Kayal-Saxena (2002)

Some problems in \mathcal{P}

\mathcal{P} . Decision problems for which there exists a poly-time algorithm.

problem	description	poly-time 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's algorithm	34, 39	34, 51
PRIMES	Is x prime?	Agrawal-Kayal-Saxena	53	51
EDIT-DISTANCE	Is the edit distance between x and y less than 5?	Needleman-Wunsch	niether neither	acgggt ttttaa
L-SOLVE	Is there a vector x that satisfies $Ax = b$?	Gauss-Edmonds elimination	$\begin{bmatrix} 0 & 1 & 1 \\ 7 & 4 & -7 \\ 0 & 3 & 15 \end{bmatrix}, \begin{bmatrix} 4 \\ 7 \\ 36 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$
U-CONN	Is an undirected graph G connected?	depth-first search		

\mathcal{NP}

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 $C(s, t) = \text{yes}$.

Def. \mathcal{NP} = set of decision 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 $p(\cdot)$.

\mathcal{NP}

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 $C(s, t) = \text{yes}$.

Def. \mathcal{NP} = set of decision 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 $p(\cdot)$.

Ex. COMPOSITES

- problem: $\{4, 6, 8, 9, 10, 12, 14, 15, 16, 18, 20, \dots\}$
- instance s : 437669
- certificate t : 541 (437, 669 = 541 \times 809)
- certifier $C(s, t)$: grade-school division

Certifiers and certificates: satisfiability

SAT. Given a CNF formula Φ , does it have a satisfying truth assignment?

3-SAT. SAT where each clause contains exactly 3 literals.

Certificate. An assignment of truth values to the Boolean variables.

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

Ex.

- instance s : $\Phi = (x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_4)$
- certificate t : $x_1 = \text{true}, x_2 = \text{true}, x_3 = \text{false}, x_4 = \text{false}$

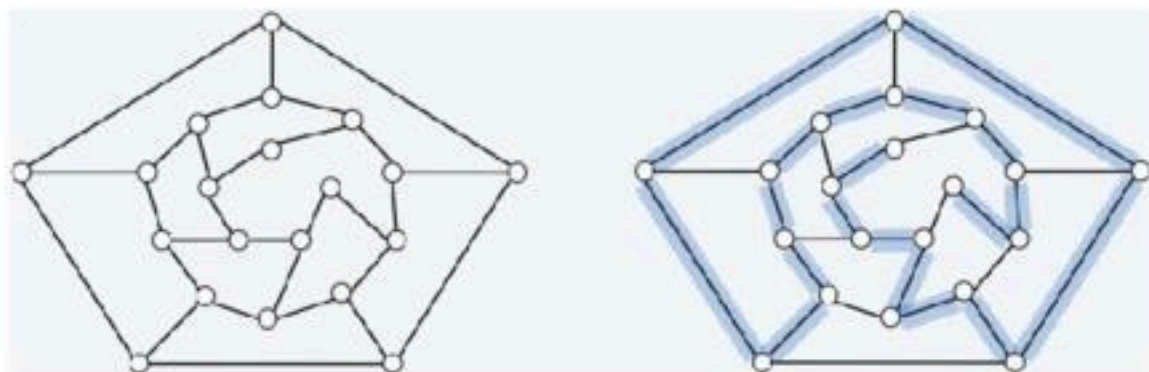
Conclusions. $\text{SAT} \in \mathcal{NP}$, $3\text{-SAT} \in \mathcal{NP}$.

Certifiers and certificates: Hamilton path

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

Certificate. A permutation π of the n nodes.


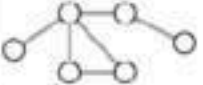
Certifier. Check that π contains each node in V exactly once, and that G contains an edge between each pair of adjacent nodes.



Conclusion. HAMILTON-PATH $\in \mathcal{NP}$.

Some problems in \mathcal{NP}

\mathcal{NP} . Decision problems for which there exists a poly-time certifier.

problem	description	poly-time 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?	Agrawal-Kayal-Saxena	51	53
FACTOR	Does x have a nontrivial factor less than y ?	???	(56159, 50)	(55687, 50)
SAT	Given a CNF formula, does it have a satisfying truth assignment?	???	$\neg x_1 \vee x_2 \vee \neg x_3$ $x_1 \vee \neg x_2 \vee x_3$ $\neg x_1 \vee \neg x_2 \vee x_3$	$\neg x_2$ $x_1 \vee x_2$ $\neg x_1 \vee x_2$
HAMILTON-PATH	Is there a simple path between u and v that visits every node?	???		

\mathcal{P} , \mathcal{NP} , and $\mathcal{EXPTIME}$

\mathcal{P} . Decision problems for which there exists a poly-time algorithm.

\mathcal{NP} . Decision problems for which there exists a poly-time certifier.

$\mathcal{EXPTIME}$. Decision problems for which there exists an exponential-time algorithm.

\mathcal{P} , \mathcal{NP} , and $\mathcal{EXPTIME}$

\mathcal{P} . Decision problems for which there exists a poly-time algorithm.

\mathcal{NP} . Decision problems for which there exists a poly-time certifier.

$\mathcal{EXPTIME}$. Decision problems for which there exists an exponential-time algorithm.

Proposition. $\mathcal{P} \subseteq \mathcal{NP}$.

Pf. Consider any problem $X \in \mathcal{P}$.

- By definition, there exists a poly-time algorithm $A(s)$ that solves X .
- Certificate $t = \phi$, certifier $C(s, t) = A(s)$.

\mathcal{P} , \mathcal{NP} , and \mathcal{EXP}

\mathcal{P} . Decision problems for which there exists a poly-time algorithm.

\mathcal{NP} . Decision problems for which there exists a poly-time certifier.

\mathcal{EXP} . Decision problems for which there exists an exponential-time algorithm.

Proposition. $\mathcal{P} \subseteq \mathcal{NP}$.

Pf. Consider any problem $X \in \mathcal{P}$.

- By definition, there exists a poly-time algorithm $A(s)$ that solves X .
- Certificate $t = \phi$, certifier $C(s, t) = A(s)$.

Proposition. $\mathcal{NP} \subseteq \mathcal{EXP}$.

Pf. Consider any problem $X \in \mathcal{NP}$.

- By definition, there exists a poly-time certifier $C(s, t)$ for X , where certificate t satisfies $|t| \leq p(|s|)$ for some polynomial $p(\cdot)$.
- To solve instance s , run $C(s, t)$ on all strings t with $|t| \leq p(|s|)$.
- Return *yes* iff $C(s, t)$ returns *yes* for any of these potential certificates.

\mathcal{P} , \mathcal{NP} , and \mathcal{EXP} (cont.)

\mathcal{P} . Decision problems for which there exists a poly-time algorithm.

\mathcal{NP} . Decision problems for which there exists a poly-time certifier.

\mathcal{EXP} . Decision problems for which there exists an exponential-time algorithm.

Proposition. $\mathcal{P} \subseteq \mathcal{NP}$.

Proposition. $\mathcal{NP} \subseteq \mathcal{EXP}$.

Fact. $\mathcal{P} \neq \mathcal{EXP} \Rightarrow$ either $\mathcal{P} \neq \mathcal{NP}$, or $\mathcal{NP} \neq \mathcal{EXP}$, or both.

The main question: \mathcal{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?

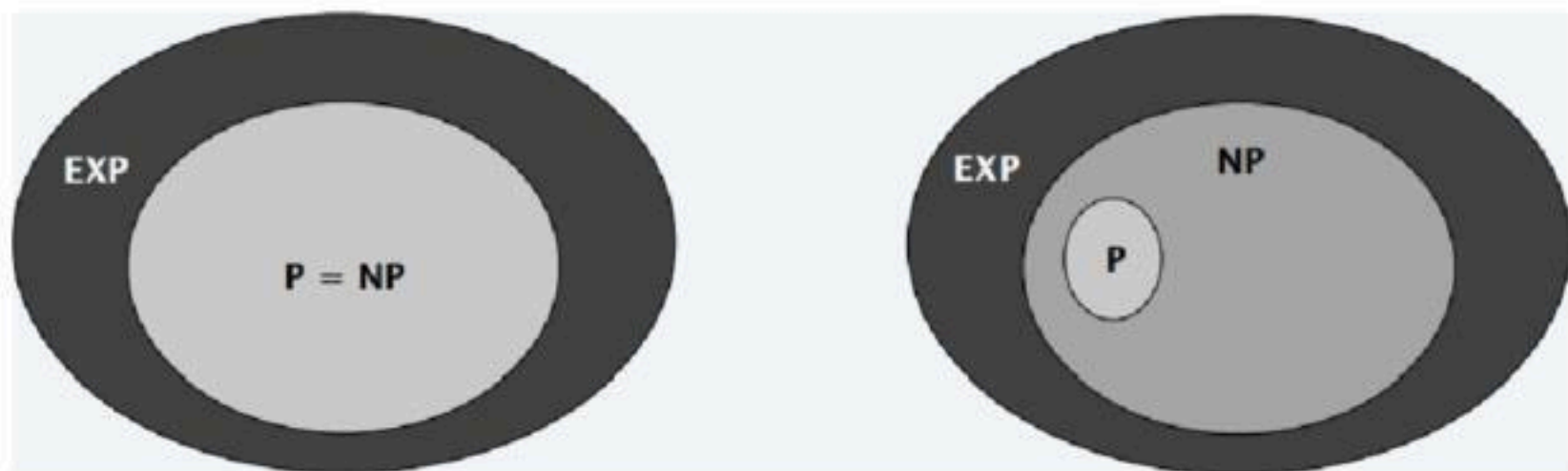
Conjecture. No poly-time algorithm (ie. *intractable*) for 3-SAT.



www.kenyon.co.uk

The main question: \mathcal{P} vs. \mathcal{NP}

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



- **If yes:** Efficient algorithms for 3-SAT, TSP, VERTEX-COVER, FACTOR, etc.
- **If no:** No efficient algorithms possible for 3-SAT, TSP, VERTEX-COVER, etc.

Consensus opinion. Probably no.

Possible outcomes: $\mathcal{P} \neq \mathcal{NP}$

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

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 \mathcal{P} is not equal to \mathcal{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 (2002)

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 $\mathcal{P} = \mathcal{NP}$ 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 (2002)

Possible outcomes: $\mathcal{P} = \mathcal{NP}$

I think that in this respect I am on the loony fringe of the mathematical community: I think (not too strongly!) that $\mathcal{P} = \mathcal{NP}$ and this will be proved within twenty years. Some years ago, Charles Read and I worked on it quite bit, and we even had a celebratory dinner in a good restaurant before we found an absolutely fatal mistake. — Béla Bollobás (2002)

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

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

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

$\mathcal{P} = \mathcal{NP}$ 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 $\mathcal{P} = \mathcal{NP}$ because there are only finitely many obstructions to the opposite hypothesis; hence there 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 $\mathcal{P} \neq \mathcal{NP}$ problem.

- The Millennium Prize Problems are seven of the most well-known and important unsolved problems in mathematics.
 - <https://brilliant.org/wiki/millennium-prize-problems/>
- The prizes were announced at a meeting in Paris, held on May 24, 2000 at the Collège de France.
 - <https://www.claymath.org/millennium-problems/millennium-prize-problems>

Princeton CS Building



binary	ASCII	char
1010000	80	P
0111101	61	=
1001110	78	N
1010000	80	P
0111111	63	?

\mathcal{NP} -complete

Polynomial transformations

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 .

Polynomial transformations

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 instance x of X , we can construct an instance y of 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.

Polynomial transformations

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 instance x of X , we can construct an instance y of 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 \mathcal{NP} ?

\mathcal{NP} -complete

\mathcal{NP} -complete. A problem $Y \in \mathcal{NP}$ with the property that for every problem $X \in \mathcal{NP}$, $X \leq_P Y$.

- “hardest” \mathcal{NP} problem.

\mathcal{NP} -complete

\mathcal{NP} -complete. A problem $Y \in \mathcal{NP}$ with the property that for every problem $X \in \mathcal{NP}$, $X \leq_P Y$.

- “hardest” \mathcal{NP} problem.

Proposition. Suppose $Y \in \mathcal{NP}$ -complete. Then, $Y \in \mathcal{P}$ iff $\mathcal{P} = \mathcal{NP}$.

Pf. \Leftarrow If $\mathcal{P} = \mathcal{NP}$, then $Y \in \mathcal{P}$ because $Y \in \mathcal{NP}$.

Pf. \Rightarrow Suppose $Y \in \mathcal{P}$.

- Consider any problem $X \in \mathcal{NP}$. Since $X \leq_P Y$, we have $X \in \mathcal{P}$.
- This implies $\mathcal{NP} \subseteq \mathcal{P}$.
- We already know $\mathcal{P} \subseteq \mathcal{NP}$. Thus $\mathcal{P} = \mathcal{NP}$.

\mathcal{NP} -complete

\mathcal{NP} -complete. A problem $Y \in \mathcal{NP}$ with the property that for every problem $X \in \mathcal{NP}$, $X \leq_P Y$.

- “hardest” \mathcal{NP} problem.

Proposition. Suppose $Y \in \mathcal{NP}$ -complete. Then, $Y \in \mathcal{P}$ iff $\mathcal{P} = \mathcal{NP}$.

Pf. \Leftarrow If $\mathcal{P} = \mathcal{NP}$, then $Y \in \mathcal{P}$ because $Y \in \mathcal{NP}$.

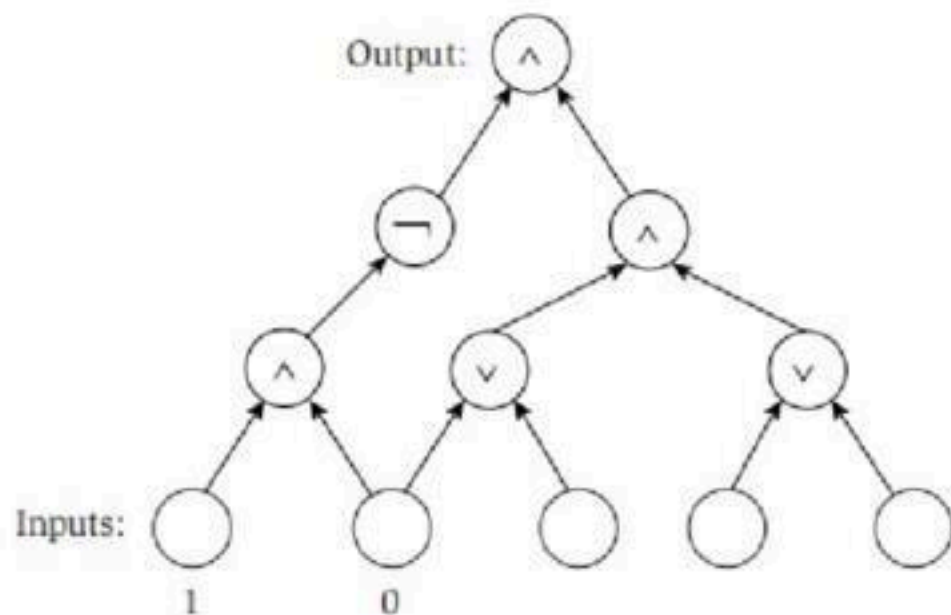
Pf. \Rightarrow Suppose $Y \in \mathcal{P}$.

- Consider any problem $X \in \mathcal{NP}$. Since $X \leq_P Y$, we have $X \in \mathcal{P}$.
- This implies $\mathcal{NP} \subseteq \mathcal{P}$.
- We already know $\mathcal{P} \subseteq \mathcal{NP}$. Thus $\mathcal{P} = \mathcal{NP}$.

Fundamental question. Are there any “natural” \mathcal{NP} -complete problems?

The “first” \mathcal{NP} -complete problem

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



Establishing NP-completeness

Remark. Once we establish first “natural” \mathcal{NP} -complete problem, others fall like dominoes.

Establishing NP-completeness

Remark. Once we establish first “natural” \mathcal{NP} -complete problem, others fall like dominoes.

Recipe. To prove that $Y \in \mathcal{NP}$ -complete:

1. Show that $Y \in \mathcal{NP}$.
2. Choose an \mathcal{NP} -complete problem X .
3. Prove that $X \leq_P Y$.

Establishing NP-completeness

Remark. Once we establish first “natural” \mathcal{NP} -complete problem, others fall like dominoes.

Recipe. To prove that $Y \in \mathcal{NP}$ -complete:

1. Show that $Y \in \mathcal{NP}$.
2. Choose an \mathcal{NP} -complete problem X .
3. Prove that $X \leq_P Y$.

Proposition. If $X \in \mathcal{NP}$ -complete, $Y \in \mathcal{NP}$, and $X \leq_P Y$, then $Y \in \mathcal{NP}$ -complete.

Pf. Consider any problem $W \in \mathcal{NP}$. Then, both $W \leq_P X$ and $X \leq_P Y$.

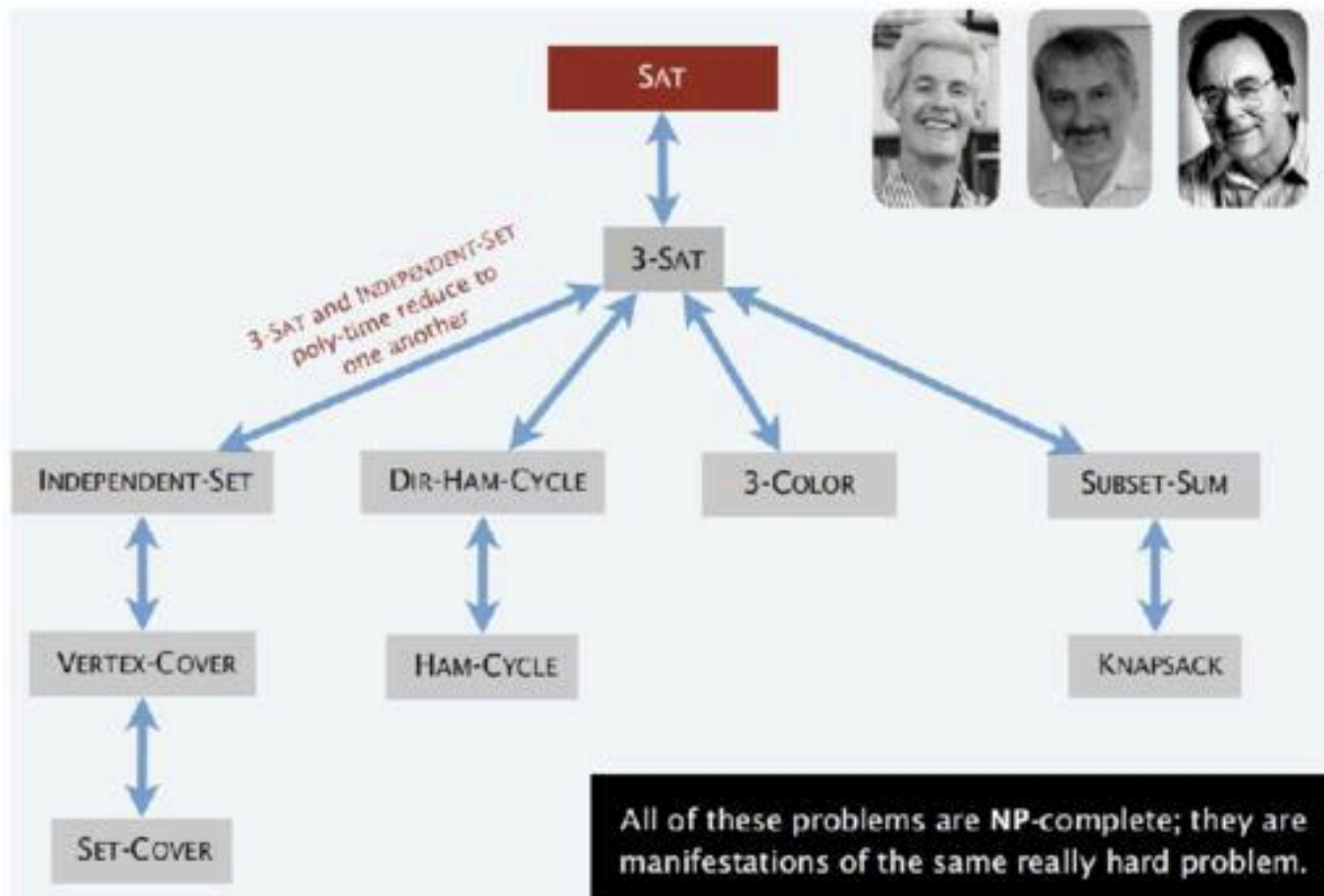
- By transitivity, $W \leq_P Y$.
- Hence $Y \in \mathcal{NP}$ -complete.

Quiz: \mathcal{NP} -complete

Suppose that $X \in \mathcal{NP}$ -complete, $Y \in \mathcal{NP}$, and $X \leq_P Y$. Which can you infer?

- A. Y is \mathcal{NP} -complete.
- B. If $Y \notin \mathcal{P}$, then $\mathcal{P} \neq \mathcal{NP}$.
- C. If $\mathcal{P} \neq \mathcal{NP}$, then neither X nor Y is in \mathcal{P} .
- D. All of the above.

Implications of Karp + Cook-Levin



Some \mathcal{NP} -complete problems

Basic genres of \mathcal{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, KNAPSACK.

Practice. Most \mathcal{NP} problems are known to be either in \mathcal{P} or \mathcal{NP} -complete.

Some \mathcal{NP} -complete problems

Basic genres of \mathcal{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, KNAPSACK.

Practice. Most \mathcal{NP} problems are known to be either in \mathcal{P} or \mathcal{NP} -complete.

\mathcal{NP} -intermediate? FACTOR, DISCRETE-LOG, GRAPH-ISOMORPHISM, etc.

Theorem. [Ladner 1975] Unless $\mathcal{P} = \mathcal{NP}$, there exist problems in \mathcal{NP} that are neither in \mathcal{P} nor \mathcal{NP} -complete.

More hard computational problems

Garey and Johnson. Computers and Intractability.

- Appendix includes over 300 \mathcal{NP} -complete problems.
- Most cited reference in computer science literature.

co-*NP*

Asymmetry of \mathcal{NP} : ex 1

Asymmetry of \mathcal{NP} . We need short certificates only for *yes* instances.

Asymmetry of \mathcal{NP} : ex 1

Asymmetry of \mathcal{NP} . We need short certificates only for *yes* instances.

Ex 1. SAT vs. UN-SAT.

- Can prove a CNF formula is satisfiable by specifying an assignment.
- How could we prove that a formula is not satisfiable?

Asymmetry of \mathcal{NP} : ex 1

Asymmetry of \mathcal{NP} . We need short certificates only for *yes* instances.

Ex 1. SAT vs. UN-SAT.

- Can prove a CNF formula is satisfiable by specifying an assignment.
- How could we prove that a formula is not satisfiable?

SAT. Given a CNF formula Φ , is there *a* satisfying truth assignment?

UN-SAT. Given a CNF formula Φ , is there *no* satisfying truth assignment?

Asymmetry of \mathcal{NP} : ex 2

Asymmetry of \mathcal{NP} . We need short certificates only for *yes* instances.

Ex 2. HAM-CYCLE vs. NO-HAM-CYCLE.

- Can prove a graph is Hamiltonian by specifying a permutation.
- How could we prove that a graph is not Hamiltonian?

Asymmetry of \mathcal{NP} : ex 2

Asymmetry of \mathcal{NP} . We need short certificates only for *yes* instances.

Ex 2. HAM-CYCLE vs. NO-HAM-CYCLE.

- Can prove a graph is Hamiltonian by specifying a permutation.
- How could we prove that a graph is not Hamiltonian?

HAM-CYCLE. Given a graph $G = (V, E)$, is there *a* simple cycle Γ that contains every node in V ?

NO-HAM-CYCLE. Given a graph $G = (V, E)$, is there *no* simple cycle Γ that contains every node in V ?

Asymmetry of \mathcal{NP} : question

Asymmetry of \mathcal{NP} . We need short certificates only for *yes* instances.

Q. How to classify UN-SAT and NO-HAM-CYCLE?

Asymmetry of \mathcal{NP} : question

Asymmetry of \mathcal{NP} . We need short certificates only for *yes* instances.

Q. How to classify UN-SAT and NO-HAM-CYCLE?

- $\text{SAT} \in \mathcal{NP}$ -complete and $\text{SAT} \equiv_P \text{UN-SAT}$.
- $\text{HAM-CYCLE} \in \mathcal{NP}$ -complete and $\text{HAM-CYCLE} \equiv_P \text{NO-HAM-CYCLE}$.
- But neither UN-SAT nor NO-HAM-CYCLE are known to be in \mathcal{NP} .

\mathcal{NP} and $\text{co-}\mathcal{NP}$

\mathcal{NP} . Decision problems for which there is a poly-time certifier.

Ex. SAT, HAM-CYCLE, and COMPOSITES.

\mathcal{NP} and $\text{co-}\mathcal{NP}$

\mathcal{NP} . Decision problems for which there is a poly-time certifier.

Ex. SAT, HAM-CYCLE, and COMPOSITES.

Def. Given a decision problem X , its **complement** \bar{X} is the same problem with *yes* and *no* answers reversed.

Ex. $X = \{4, 6, 8, 9, 10, 12, 14, 15, \dots\}$

- $\bar{X} = 2, 3, 5, 7, 11, 13, 17, 23, 29, \dots$
 - note: ignore 0 and 1 (neither prime nor composite)

\mathcal{NP} and $\text{co-}\mathcal{NP}$

\mathcal{NP} . Decision problems for which there is a poly-time certifier.

Ex. SAT, HAM-CYCLE, and COMPOSITES.

Def. Given a decision problem X , its **complement** \bar{X} is the same problem with *yes* and *no* answers reversed.

Ex. $X = \{4, 6, 8, 9, 10, 12, 14, 15, \dots\}$

- $\bar{X} = 2, 3, 5, 7, 11, 13, 17, 23, 29, \dots$
 - note: ignore 0 and 1 (neither prime nor composite)

$\text{co-}\mathcal{NP}$. Complements of decision problems in \mathcal{NP} .

Ex. UN-SAT, NO-HAM-CYCLE, and PRIMES.

$\mathcal{NP} = \text{co-NP}?$

Fundamental open question. Does $\mathcal{NP} = \text{co-NP}$?

- Do *yes* instances have succinct certificates iff *no* instances do?
- Consensus opinion: no.

$\mathcal{NP} = \text{co-NP}$?

Fundamental open question. Does $\mathcal{NP} = \text{co-NP}$?

- Do *yes* instances have succinct certificates iff *no* instances do?
- Consensus opinion: no.

Theorem. If $\mathcal{NP} \neq \text{co-NP}$, then $\mathcal{P} \neq \mathcal{NP}$.

Pf idea.

- \mathcal{P} is closed under complementation.
- If $\mathcal{P} = \mathcal{NP}$, then \mathcal{NP} is closed under complementation.
- In other words, $\mathcal{NP} = \text{co-NP}$.
- This is the contrapositive of the theorem.

Good characterizations

Good characterization. [Edmonds 1965] $\mathcal{NP} \cap \text{co-NP}$.

- If problem X is in both \mathcal{NP} and co-NP , then:
 - for *yes* instance, there is a succinct certificate
 - for *no* instance, there is a succinct disqualifier
- Provides conceptual leverage for reasoning about a problem.

Good characterizations

Good characterization. [Edmonds 1965] $\mathcal{NP} \cap \text{co-NP}$.

- If problem X is in both \mathcal{NP} and co-NP , then:
 - for *yes* instance, there is a succinct certificate
 - for *no* instance, there is a succinct disqualifier
- Provides conceptual leverage for reasoning about a problem.

Ex. Given a bipartite graph, is there a perfect matching?

- If *yes*, can exhibit a perfect matching.
- If *no*, can exhibit a set of nodes S such that $|\text{neighbors}(S)| < |S|$.
 - means: no need to enumerate all possibilities
 - note (as in paper): "... does *not* mean necessarily that there is a good algorithm."

Good characterizations: ex 1

Observation. $\mathcal{P} \subseteq \mathcal{NP} \cap \text{co-NP}$.

- Proof of max-flow min-cut theorem led to stronger result that max-flow and min-cut are in \mathcal{P} .
 - max-flow complements to min-cut
- Sometimes finding a good characterization seems easier than finding an efficient algorithm.

Good characterizations: ex 1

Observation. $\mathcal{P} \subseteq \mathcal{NP} \cap \text{co-NP}$.

- Proof of max-flow min-cut theorem led to stronger result that max-flow and min-cut are in \mathcal{P} .
 - max-flow complements to min-cut
- Sometimes finding a good characterization seems easier than finding an efficient algorithm.

Fundamental open question. Does $\mathcal{P} = \mathcal{NP} \cap \text{co-NP}$?

- Mixed opinions.
- Many examples where problem found to have a nontrivial good characterization, but only years later discovered to be in \mathcal{P} .

Linear programming is in $\mathcal{NP} \cap \text{co-NP}$

LINEAR-PROGRAMMING. Given $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, $c \in \mathbb{R}^n$, and $\alpha \in \mathbb{R}$, does there exist $x \in \mathbb{R}^n$ such that $Ax \leq b$, $x \geq 0$ and $c^T x \geq \alpha$?

Linear programming is in $\mathcal{NP} \cap \text{co-NP}$

LINEAR-PROGRAMMING. Given $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, $c \in \mathbb{R}^n$, and $\alpha \in \mathbb{R}$, does there exist $x \in \mathbb{R}^n$ such that $Ax \leq b$, $x \geq 0$ and $c^T x \geq \alpha$?

Theorem. [Gale-Kuhn-Tucker 1948] $\text{LINEAR-PROGRAMMING} \in \mathcal{NP} \cap \text{co-NP}$.

Pf sketch. If (P) and (D) are nonempty, then $\max = \min$.

$$\text{(Primary) } \max c^T x$$

$$\text{s.t. } Ax \leq b$$

$$x \geq 0$$

$$\text{(Dual) } \min y^T b$$

$$\text{s.t. } A^T y \geq c$$

$$y \geq 0$$

Linear programming is in $\mathcal{NP} \cap \text{co-NP}$

LINEAR-PROGRAMMING. Given $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, $c \in \mathbb{R}^n$, and $\alpha \in \mathbb{R}$, does there exist $x \in \mathbb{R}^n$ such that $Ax \leq b$, $x \geq 0$ and $c^T x \geq \alpha$?

Theorem. [Gale-Kuhn-Tucker 1948] $\text{LINEAR-PROGRAMMING} \in \mathcal{NP} \cap \text{co-NP}$.

Pf sketch. If (P) and (D) are nonempty, then $\max = \min$.

$$\text{(Primary) } \max c^T x$$

$$\text{s.t. } Ax \leq b$$

$$x \geq 0$$

$$\text{(Dual) } \min y^T b$$

$$\text{s.t. } A^T y \geq c$$

$$y \geq 0$$

Theorem. [Khachiyan 1979] $\text{LINEAR-PROGRAMMING} \in \mathcal{P}$.

Primality testing is in $\mathcal{NP} \cap \text{co-NP}$

Theorem. [Pratt 1975] $\text{PRIMES} \in \mathcal{NP} \cap \text{co-NP}$.

Pf sketch. An odd integer s is prime iff there exists an integer $1 < t < s$ s.t.

$$t^{s-1} \equiv 1 \pmod{s}$$

$$t^{\frac{s-1}{p}} \not\equiv 1 \pmod{s}$$

for all prime divisors p of $s - 1$.

- instance s : 437677
- certificate t : $17, 2^2 \times 3 \times 36473$
 - prime factorization of $s - 1$ also need a recursive certificate to assert 36,473 is prime

CERTIFIER (s)

CHECK $s - 1 = 2 \times 2 \times 3 \times 36473$.

CHECK $17^{s-1} \equiv 1 \pmod{s}$.

CHECK $17^{(s-1)/2} \equiv 437676 \pmod{s}$.

CHECK $17^{(s-1)/3} \equiv 329415 \pmod{s}$.

CHECK $17^{(s-1)/36,473} \equiv 305452$

\pmod{s} .

Primality testing is in \mathcal{P}

Theorem. [Agrawal-Kayal-Saxena 2004] PRIMES $\in \mathcal{P}$.

Factoring is in $\mathcal{NP} \cap \text{co-NP}$

FACTORIZE. Given an integer x , *find* its prime factorization.

FACTOR. Given two integers x and y , does x have a nontrivial factor $< y$?

Factoring is in $\mathcal{NP} \cap \text{co-NP}$

FACTORIZE. Given an integer x , *find* its prime factorization.

FACTOR. Given two integers x and y , does x have a nontrivial factor $< y$?

Theorem. $\text{FACTOR} \equiv_P \text{FACTORIZE}$.

Pf.

- $\leq \mathcal{P}$ trivial.
- $\geq \mathcal{P}$ binary search to find a factor; divide out the factor and repeat.

Factoring is in $\mathcal{NP} \cap \text{co-NP}$

FACTORIZE. Given an integer x , *find* its prime factorization.

FACTOR. Given two integers x and y , does x have a nontrivial factor $< y$?

Theorem. $\text{FACTOR} \equiv_P \text{FACTORIZE}$.

Pf.

- $\leq \mathcal{P}$ trivial.
- $\geq \mathcal{P}$ binary search to find a factor; divide out the factor and repeat.

Theorem. $\text{FACTOR} \in \mathcal{NP} \cap \text{co-NP}$.

Pf.

- Certificate: a factor p of x that is less than y .
- Disqualifier: the prime factorization of x (where each prime factor is less than y), along with a Pratt certificate that each factor is prime.

Is factoring in \mathcal{P} ?

Fundamental question. Is FACTOR $\in \mathcal{P}$?

Is factoring in \mathcal{P} ?

Fundamental question. Is FACTOR $\in \mathcal{P}$?

Challenge. [RSA-704] Factor this number.

74037563479561712828046796097429573142593188889231289
08493623263897276503402826627689199641962511784399589
43305021275853701189680982867331732731089309005525051
16877063299072396380786710086096962537934650563796359

(\$30,000 prize if you can factor)

Exploiting intractability

Modern cryptography.

- **Ex.** Send your credit card number to Amazon.
- **Ex.** Digitally sign an e-document.
- Enables freedom of privacy, speech, press, etc.

Exploiting intractability

Modern cryptography.

- **Ex.** Send your credit card number to Amazon.
- **Ex.** Digitally sign an e-document.
- Enables freedom of privacy, speech, press, etc.

RSA. Based on dichotomy between complexity of two problems.

- To use: generate two random n -bit primes and multiply.
- To break: suffices to factor a $2n$ -bit integer.

Factoring on a quantum computer

Theorem. [Shor 1994] Can factor an n -bit integer in $O(n^3)$ steps on a “quantum computer.”

2001. Factored $15 = 3 \times 5$ (with high probability) on a quantum computer.

2012. Factored $21 = 3 \times 7$.

Factoring on a quantum computer

Theorem. [Shor 1994] Can factor an n -bit integer in $O(n^3)$ steps on a “quantum computer.”

2001. Factored $15 = 3 \times 5$ (with high probability) on a quantum computer.

2012. Factored $21 = 3 \times 7$.

Fundamental question. Does $\mathcal{P} = \mathcal{BQP}$?

- quantum analog of \mathcal{P} (bounded error quantum polynomial time)

NP-hard

A note on terminology

A Terminological Proposal. [Knuth 1974]

I needed an adjective to convey such a degree of difficulty, both formally and informally; ... The goal is to find an adjective x that sounds good in sentences like this:

- *The covering problem is x .*
- *It is x to decide whether a given graph has a Hamiltonian circuit.*
- *It is unknown whether or not primality testing is an x problem.*

A note on terminology

A Terminological Proposal. [Knuth 1974]

I needed an adjective to convey such a degree of difficulty, both formally and informally; ... The goal is to find an adjective x that sounds good in sentences like this:

- *The covering problem is x .*
- *It is x to decide whether a given graph has a Hamiltonian circuit.*
- *It is unknown whether or not primality testing is an x problem.*

Note. The term x does not necessarily imply that a problem is in \mathcal{NP} , just that every problem in \mathcal{NP} poly-time reduces to x .

Terminology suggestions

Knuth's original suggestions.

- Hard, Tough: too common and may already been used.
- Herculean, Formidable, Arduous: arcane.

Terminology suggestions

Knuth's original suggestions.

- Hard, Tough: too common and may already been used.
- Herculean, Formidable, Arduous: arcane.

Some English word write-ins.

- Impractical.
- Bad.
- Heavy.
- Tricky.
- Intricate.
- Prodigious.
- Difficult.
- Intractable.
- Costly.
- Obdurate.
- Obstinate.

- Exorbitant.
- Interminable.

Terminology in literature

Hard-boiled. [Ken Steiglitz] In honor of Cook.

Hard-ass. [Al Meyer] Hard AS Satisfiability.

Sisyphean. [Bob Floyd] Problem of Sisyphus was time-consuming.

Ulyssean. [Donald Knuth] Ulysses was known for his persistence.

Terminology in literature

Hard-boiled. [Ken Steiglitz] In honor of Cook.

Hard-ass. [Al Meyer] Hard AS Satisfiability.

Sisyphean. [Bob Floyd] Problem of Sisyphus was time-consuming.

Ulyssean. [Donald Knuth] Ulysses was known for his persistence.

“ creative research workers are as full of ideas for new terminology as they are empty of enthusiasm for adopting it. ” — Donald Knuth

Terminology: acronyms

PET. [Shen Lin] Probably Exponential Time.

- If $\mathcal{P} \neq \mathcal{NP}$, Provably Exponential Time.
- If $\mathcal{P} = \mathcal{NP}$, Previously Exponential Time.

GNP. [Al Meyer] Greater than or equal to \mathcal{NP} in difficulty.

- And costing more than the GNP to solve.

Terminology: made-up words

Exparent. [Mike Paterson] Exponential + apparent.

Perarduous. [Mike Paterson] Throughout (in space or time) + completely.

Supersat. [Al Meyer] Greater than or equal to satisfiability.

Polychronious. [Ed Reingold] Enduringly long; chronic.

Terminology: consensus

\mathcal{NP} -complete. A problem in \mathcal{NP} such that every problem in \mathcal{NP} poly-time reduces to it.

\mathcal{NP} -hard. [Bell Labs, Steve Cook, Ron Rivest, Sartaj Sahni]
A problem such that every problem in \mathcal{NP} poly-time reduces to it.

Terminology: consensus

\mathcal{NP} -complete. A problem in \mathcal{NP} such that every problem in \mathcal{NP} poly-time reduces to it.

\mathcal{NP} -hard. [Bell Labs, Steve Cook, Ron Rivest, Sartaj Sahni]
A problem such that every problem in \mathcal{NP} poly-time reduces to it.

“If the Martians know that $\mathcal{P} = \mathcal{NP}$ for Turing Machines and they kidnap me, I would lose face calling these problems ‘formidable’.” — Vaughan Pratt.