Classify problems according to computational requirements

Q. Which problems will we be able to solve in practice?

A working definition. Those with polynomial-time algorithms.

Theory. Definition is broad and robust.

Practice. Poly-time algorithms scale to huge problems.

\[ \text{constants tend to be small, e.g., } 3n^2 \]

A is \(\text{NP-complete}: \)

\[ \begin{align*}
(1) & \text{ } A \in \text{NP} \\
(2) & \text{ } A \leq_P \text{ B } \in \text{NP}, \text{ B } \leq_P \text{ A}
\end{align*} \]

Factoring \(\text{NP} \) is \(\text{NP-complete} \)

\[ \text{ decision problem: } \text{Yes} \text{ No} \]

NP: with \& short easy \& check witnesses for Yes

\[ \text{SAT: } \phi(x_1, \ldots, x_n) \]

\[ \exists \phi \in \text{SAT}? \]
Polynomial-time reductions

Desiderata'. Suppose we could solve problem $Y$ in polynomial-time. What else could we solve in polynomial time?

Reduction. Problem $X$ polynomial-time (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 $Y$.

[Diagram showing the reduction process with labeled nodes: instance I (of X), Algorithm for X, Algorithm for Y, solution S to I.]

Computational model supplemented by special piece of hardware that solves instances of $Y$ in a single step.
**Polynomial-time reductions**

**Desiderata.** Suppose we could solve problem $Y$ in polynomial-time. What else could we solve in polynomial time?

**Reduction.** Problem $X$ **polynomial-time (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 $Y$.

**Notation.** $X \leq_p Y$.

**Note.** We pay for time to write down instances sent to oracle $\Rightarrow$ instances of $Y$ must be of polynomial size.

**Caveat.** Don’t mistake $X \leq_p Y$ with $Y \leq_p X$.

$X, Y$ **NP-complete**: $X \leq_p Y \& Y \leq_p X$
Polynomial-time reductions

**Design algorithms.** If $X \leq_p Y$ and $Y$ can be solved in polynomial time, then $X$ can be solved in polynomial time.

**Establish intractability.** If $X \leq_p Y$ and $X$ cannot be solved in polynomial time, then $Y$ cannot be solved in polynomial time.

**Establish equivalence.** If both $X \leq_p Y$ and $Y \leq_p X$, we use notation $X \equiv_p Y$. In this case, $X$ can be solved in polynomial time iff $Y$ can be.

**Bottom line.** Reductions classify problems according to relative difficulty.
**Independent set**

**INDEPENDENT-SET.** Given a graph $G = (V, E)$ and an integer $k$, is there a subset of vertices $S \subseteq V$ such that $|S| \geq k$, and for each edge at most one of its endpoints is in $S$?

**Ex.** Is there an independent set of size $\geq 6$?

**Ex.** Is there an independent set of size $\geq 7$?

![Diagram of a graph with an independent set of size 6 highlighted]
Vertex cover

**Vertex-Cover.** Given a graph $G = (V, E)$ and an integer $k$, is there a subset of vertices $S \subseteq V$ such that $|S| \leq k$, and for each edge, at least one of its endpoints is in $S$?

**Ex.** Is there a vertex cover of size $\leq 4$?
**Ex.** Is there a vertex cover of size $\leq 3$?
Vertex cover and independent set reduce to one another

**Theorem.** $\text{VERTEX-COVER} \equiv_p \text{INDEPENDENT-SET}.$

**Pf.** We show $S$ is an independent set of size $k$ iff $\overline{V - S}$ is a vertex cover of size $n - k.$
Vertex cover and independent set reduce to one another

**Theorem.** \( \text{VERTEX-COVER} \equiv \text{p} \ \text{INDEPENDENT-SET}. \)

**Pf.** We show \( S \) is an independent set of size \( k \) iff \( V - S \) is a vertex cover of size \( n - k \).

\[ \Rightarrow \]

- Let \( S \) be any independent set of size \( k \).
- \( V - S \) is of size \( n - k \).
- Consider an arbitrary edge \((u, v)\).
- \( S \) independent \( \Rightarrow \) either \( u \notin S \) or \( v \notin S \) (or both)
  \[ \Rightarrow \] either \( u \in V - S \) or \( v \in V - S \) (or both).
- Thus, \( V - S \) covers \((u, v)\).
Vertex cover and independent set reduce to one another

**Theorem.** \( \text{VERTEX-COVER} \equiv_p \text{INDEPENDENT-SET}. \)

**Pf.** We show \( S \) is an independent set of size \( k \) iff \( V - S \) is a vertex cover of size \( n - k \).

\[ \leftarrow \]

- Let \( V - S \) be any vertex cover of size \( n - k \).
- \( S \) is of size \( k \).
- Consider two nodes \( u \in S \) and \( v \in S \).
- Observe that \( (u, v) \notin E \) since \( V - S \) is a vertex cover.
- Thus, no two nodes in \( S \) are joined by an edge \( \Rightarrow S \) independent set. \( \blacksquare \)
Set cover

**Set-Cover.** Given a set $U$ of elements, a collection $S$ of subsets of $U$, and an integer $k$, are there $\leq k$ of these subsets whose union is equal to $U$?

Sample application.

- $m$ available pieces of software.
- Set $U$ of $n$ capabilities that we would like our system to have.
- The $i^{th}$ piece of software provides the set $S_i \subseteq U$ of capabilities.
- Goal: achieve all $n$ capabilities using fewest pieces of software.

\[
\begin{align*}
U &= \{1, 2, 3, 4, 5, 6, 7\} \\
S_a &= \{3, 7\} & S_b &= \{2, 4\} \\
\textcolor{blue}{S_c} &= \{3, 4, 5, 6\} & S_d &= \{5\} \\
S_e &= \{1\} & \textcolor{blue}{S_f} &= \{1, 2, 6, 7\} \\
k &= 2
\end{align*}
\]

a set cover instance
Vertex cover reduces to set cover

Theorem. \( \text{VERTEX-COVER} \leq_p \text{SET-COVER} \).

Pf. Given a \( \text{VERTEX-COVER} \) instance \( G = (V, E) \) and \( k \), we construct a \( \text{SET-COVER} \) instance \( (U, S) \) that has a set cover of size \( k \) iff \( G \) has a vertex cover of size \( k \).

Construction.
- Universe \( U = E \).
- Include one subset for each node \( v \in V : S_v = \{ e \in E : e \text{ incident to } v \} \).

\[
\begin{align*}
U &= \{ 1, 2, 3, 4, 5, 6, 7 \} \\
S_a &= \{ 3, 7 \} & S_b &= \{ 2, 4 \} \\
S_c &= \{ 3, 4, 5, 6 \} & S_d &= \{ 5 \} \\
S_e &= \{ 1 \} & S_f &= \{ 1, 2, 6, 7 \}
\end{align*}
\]
Vertex cover reduces to set cover

**Lemma.** $G = (V, E)$ contains a vertex cover of size $k$ iff $(U, S)$ contains a set cover of size $k$.

**Pf.** $\Rightarrow$ Let $X \subseteq V$ be a vertex cover of size $k$ in $G$.

- Then $Y = \{ S_v : v \in X \}$ is a set cover of size $k$. ■

---

**Vertex cover instance**

$(k = 2)$

$G = (V, E)$ contains a vertex cover of size $k$.

$U = \{1, 2, 3, 4, 5, 6, 7\}$

$S_a = \{3, 7\}$

$S_b = \{2, 4\}$

$S_c = \{3, 4, 5, 6\}$

$S_d = \{5\}$

$S_e = \{1\}$

$S_f = \{1, 2, 6, 7\}$

---

**Set cover instance**

$(k = 2)$

$(k = 2)$
Vertex cover reduces to set cover

**Lemma.** $G = (V, E)$ contains a vertex cover of size $k$ iff $(U, S)$ contains a set cover of size $k$.

**Pf.** $\Leftarrow$ Let $Y \subseteq S$ be a set cover of size $k$ in $(U, S)$.
- Then $X = \{ v : S_v \in Y \}$ is a vertex cover of size $k$ in $G$. $\blacksquare$

\[ U = \{ 1, 2, 3, 4, 5, 6, 7 \} \]
\[ S_a = \{ 3, 7 \} \]
\[ S_c = \{ 2, 4 \} \]
\[ S_e = \{ 3, 4, 5, 6 \} \]
\[ S_f = \{ 5 \} \]
\[ S_v = \{ 1 \} \]

\[ S_x = \{ 1, 2, 6, 7 \} \]

vertex cover instance  
(k = 2)  
set cover instance  
(k = 2)
Satisfiability

Literal. A Boolean variable or its negation. \( x_i \) or \( \overline{x_i} \)

Clause. A disjunction of literals. \( C_j = x_1 \lor x_2 \lor x_3 \)

Conjunctive normal form (CNF). A propositional formula \( \Phi \) that is a conjunction of clauses.

\[ \Phi = C_1 \land C_2 \land C_3 \land C_4 \]

SAT. Given a CNF formula \( \Phi \), does it have a satisfying truth assignment?

3-SAT. SAT where each clause contains exactly 3 literals (and each literal corresponds to a different variable).

\[ \Phi = \left( \overline{x_1} \lor x_2 \lor x_3 \right) \land \left( x_1 \lor \overline{x_2} \lor x_3 \right) \land \left( \overline{x_1} \lor x_2 \lor \overline{x_3} \right) \]

yes instance: \( x_1 = \text{true}, x_2 = \text{true}, x_3 = \text{false}, x_4 = \text{false} \)

Key application. Electronic design automation (EDA).
3-satisfiability reduces to independent set

**Theorem.** $\text{3-SAT} \leq_p \text{INDEPENDENT-SET}.$

**Pf.** Given an instance $\Phi$ of 3-SAT, we construct an instance $(G, k)$ of \text{INDEPENDENT-SET} that has an independent set of size $k = |\Phi|$ iff $\Phi$ is satisfiable.

**Construction.**
- $G$ contains 3 nodes for each clause, one for each literal.
- Connect 3 literals in a clause in a triangle.
- Connect literal to each of its negations.

$$\Phi = (\overline{x}_1 \lor x_2 \lor x_3) \land (x_1 \lor \overline{x}_2 \lor x_3) \land (\overline{x}_1 \lor x_2 \lor x_4)$$
Lemma. $G$ contains independent set of size $k = |\Phi|$ iff $\Phi$ is satisfiable.

Pf. $\Rightarrow$ Let $S$ be independent set of size $k$.
- $S$ must contain exactly one node in each triangle.
- Set these literals to true (and remaining variables consistently).
- Truth assignment is consistent and all clauses are satisfied.

Pf. $\Leftarrow$ Given satisfying assignment, select one true literal from each triangle. This is an independent set of size $k$. 

$k = 3$

Assume $\Phi = (\overline{x_1} \lor x_2 \lor x_3) \land (x_1 \lor \overline{x_2} \lor x_3) \land (x_1 \lor x_2 \lor x_4)$ is sat: $x_2 = 1$ and $x_1 = 1$.
Review

Basic reduction strategies.

- Simple equivalence: \textsc{Independent-Set} \equiv_p \textsc{Vertex-Cover}.
- Special case to general case: \textsc{Vertex-Cover} \leq_p \textsc{Set-Cover}.
- Encoding with gadgets: \textsc{3-Sat} \leq_p \textsc{Independent-Set}.

Transitivity. If $X \leq_p Y$ and $Y \leq_p Z$, then $X \leq_p Z$.

Pf idea. Compose the two algorithms.

\textbf{Ex.} \textsc{3-Sat} \leq_p \textsc{Independent-Set} \leq_p \textsc{Vertex-Cover} \leq_p \textsc{Set-Cover}.
Search problems

**Decision problem.** Does there exist a vertex cover of size $\leq k$?

**Search problem.** Find a vertex cover of size $\leq k$.

**Ex.** To find a vertex cover of size $\leq k$:

- Determine if there exists a vertex cover of size $\leq k$.
- Find a vertex $v$ such that $G - \{v\}$ has a vertex cover of size $\leq k - 1$.
  (any vertex in any vertex cover of size $\leq k$ will have this property)
- Include $v$ in the vertex cover.
- Recursively find a vertex cover of size $\leq k - 1$ in $G - \{v\}$.

**Bottom line.** $\text{VERTEX-COVER} \equiv_p \text{FIND-VERTEX-COVER}$.

**Exercise:** $\text{SAT} \equiv_p \text{FIND-VERTEX-COVER}$.

Optimization problems

**Decision problem.** Does there exist a vertex cover of size $\leq k$?

**Search problem.** Find a vertex cover of size $\leq k$.

**Optimization problem.** Find a vertex cover of minimum size.

**Ex.** To find vertex cover of minimum size:

- (Binary) search for size $k^*$ of min vertex cover.
- Solve corresponding search problem.

**Bottom line.** $\text{VERTEX-COVER} \equiv_p \text{FIND-VERTEX-COVER} \equiv_p \text{OPTIMAL-VERTEX-COVER}$. 
3-dimensional matching

**3D-MATCHING.** Given 3 disjoint sets $X$, $Y$, and $Z$, each of size $n$ and a set $T \subseteq X \times Y \times Z$ of triples, does there exist a set of $n$ triples in $T$ such that each element of $X \cup Y \cup Z$ is in exactly one of these triples?

\[
X = \{ x_1, x_2, x_3 \}, \quad Y = \{ y_1, y_2, y_3 \}, \quad Z = \{ z_1, z_2, z_3 \}
\]

\[
T_1 = \{ x_1, y_1, z_2 \}, \quad T_2 = \{ x_1, y_2, z_1 \}, \quad T_3 = \{ x_1, y_2, z_2 \}
\]

\[
T_4 = \{ x_2, y_2, z_3 \}, \quad T_5 = \{ x_2, y_3, z_3 \}, \quad T_6 = \{ x_2, y_2, z_3 \}
\]

\[
T_7 = \{ x_3, y_1, z_3 \}, \quad T_8 = \{ x_3, y_1, z_1 \}, \quad T_9 = \{ x_3, y_2, z_1 \}
\]

an instance of 3d–matching (with $n = 3$)

**Remark.** Generalization of bipartite matching.
3-dimensional matching

**3D-MATCHING.** Given 3 disjoint sets $X$, $Y$, and $Z$, each of size $n$ and a set $T \subseteq X \times Y \times Z$ of triples, does there exist a set of $n$ triples in $T$ such that each element of $X \cup Y \cup Z$ is in exactly one of these triples?

**Theorem.** $3$-SAT $\leq_p$ 3D-MATCHING.

**Pf.** Given an instance $\Phi$ of 3-SAT, we construct an instance of 3D-MATCHING that has a perfect matching iff $\Phi$ is satisfiable.

$2$-SAT vs. 2D Matching

3-SAT vs. 3D-Matching

NP-complete
3-colorability

3-COLOR. Given an undirected graph $G$, can the nodes be colored black, white, and blue so that no adjacent nodes have the same color?

VS.

2-COLOR (=$\text{Bipartiteness}$) $\in$ P

yes instance
Application: register allocation

Register allocation. Assign program variables to machine registers so that no more than $k$ registers are used and no two program variables that are needed at the same time are assigned to the same register.

Interference graph. Nodes are program variables; edge between $u$ and $v$ if there exists an operation where both $u$ and $v$ are "live" at the same time.

Observation. [Chaitin 1982] Can solve register allocation problem iff interference graph is $k$-colorable.

Fact. $3$-COLOR $\leq_p K$-REGISTER-ALLOCATION for any constant $k \geq 3$. 

REGISTER ALLOCATION & SPILLING VIA GRAPH COLORING
G. J. Chaitin
IBM Research
P.O. Box 218, Yorktown Heights, NY 10598
3-satisfiability reduces to 3-colorability

**Theorem.** 3-SAT $\leq_p$ 3-COLOR.

**Pf.** Given 3-SAT instance $\Phi$, we construct an instance of 3-COLOR that is 3-colorable iff $\Phi$ is satisfiable.
3-satisfiability reduces to 3-colorability

**Construction.**

(i) Create a graph \( G \) with a node for each literal.
(ii) Connect each literal to its negation.
(iii) Create 3 new nodes \( T, F, \) and \( B \); connect them in a triangle.
(iv) Connect each literal to \( B \).
(v) For each clause \( C_i \), add a gadget of 6 nodes and 13 edges.

↑

to be described later
3-satisfiability reduces to 3-colorability

Lemma. Graph $G$ is 3-colorable iff $\Phi$ is satisfiable.

Pf. $\Rightarrow$ Suppose graph $G$ is 3-colorable.

- WLOG, assume that node $T$ is colored $\text{black}$, $F$ is $\text{white}$, and $B$ is $\text{blue}$.
- Consider assignment that sets all $\text{black}$ literals to $\text{true}$ (and $\text{white}$ to $\text{false}$).
- (iv) ensures each literal is colored either $\text{black}$ or $\text{white}$.
- (ii) ensures that each literal is $\text{white}$ if its negation is $\text{black}$ (and vice versa).
3-satisfiability reduces to 3-colorability

**Lemma.** Graph $G$ is 3-colorable iff $\Phi$ is satisfiable.

**Pf.** $\Rightarrow$ Suppose graph $G$ is 3-colorable.

- WLOG, assume that node $T$ is colored black, $F$ is white, and $B$ is blue.
- Consider assignment that sets all black literals to true (and white to false).
- (iv) ensures each literal is colored either black or white.
- (ii) ensures that each literal is white if its negation is black (and vice versa).
- (v) ensures at least one literal in each clause is black.

![Diagram showing the reduction from 3-satisfiability to 3-colorability](image)

$C_j = x_1 \lor \overline{x_2} \lor x_3$
3-satisfiability reduces to 3-colorability

**Lemma.** Graph $G$ is 3-colorable iff $\Phi$ is satisfiable.

**Pf.** Suppose graph $G$ is 3-colorable.

- WLOG, assume that node $T$ is colored *black*, $F$ is *white*, and $B$ is *blue*.
- Consider assignment that sets all *black* literals to true (and *white* to false).
- (iv) ensures each literal is colored either *black* or *white*.
- (ii) ensures that each literal is *white* if its negation is *black* (and vice versa).
- (v) ensures at least one literal in each clause is *black*.

$$C_j = x_1 \lor \overline{x_2} \lor x_3$$

Contradiction (not a 3-coloring)

Suppose, for the sake of contradiction, that all 3 literals are white in some 3-coloring
3-satisfiability reduces to 3-colorability

**Lemma.** Graph $G$ is 3-colorable iff $\Phi$ is satisfiable.

**Pf.** $\Leftarrow$ Suppose 3-SAT instance $\Phi$ is satisfiable.

- Color all *true* literals *black* and all *false* literals *white*.
- Pick one *true* literal; color node below that node *white*, and node below that *blue*.
- Color remaining middle row nodes *blue*.
- Color remaining bottom nodes *black* or *white*, as forced. □

\[ C_j = x_1 \lor \overline{x_2} \lor x_3 \]

A literal set to true in 3-SAT assignment.
Polynomial-time reductions

constraint satisfaction

3-SAT

3-SAT poly-time reduces to INDEPENDENT-SET

INDEPENDENT-SET

3-SAT poly-time reduces to INDEPENDENT-SET

INDEPENDENT-SET

VERTEX-COVER

SET-COVER

packing and covering

HAM-CYCLE

TSP

sequencing

GRAPH-3-COLOR

PLANAR-3-COLOR

partitioning

SUBSET-SUM

SCHEDULING

numerical
**Subset sum**

**SUBSET-SUM.** Given natural numbers $w_1, \ldots, w_n$ and an integer $W$, is there a subset that adds up to exactly $W$?

**Ex.** $\{ 1, 4, 16, 64, 256, 1040, 1041, 1093, 1284, 1344 \}$, $W = 3754$.
**Yes.** $1 + 16 + 64 + 256 + 1040 + 1093 + 1284 = 3754$.

**Remark.** With arithmetic problems, input integers are encoded in binary. Poly-time reduction must be polynomial in binary encoding.
Subset sum

Theorem. 3-SAT $\leq_p$ SUBSET-SUM.

Proof. Given an instance $\Phi$ of 3-SAT, we construct an instance of SUBSET-SUM that has solution iff $\Phi$ is satisfiable.
### 3-satisfiability reduces to subset sum

**Construction.** Given 3-SAT instance $\Phi$ with $n$ variables and $k$ clauses, form $2n + 2k$ decimal integers, each of $n + k$ digits:

- Include one digit for each variable $x_i$ and for each clause $C_j$.
- Include two numbers for each variable $x_i$.
- Include two numbers for each clause $C_j$.
- Sum of each $x_i$ digit is 1; sum of each $C_j$ digit is 4.

**Key property.** No carries possible $\Rightarrow$ each digit yields one equation.

![3-SAT instance diagram]

\[
\begin{aligned}
C_1 &= \neg x_1 \lor x_2 \lor x_3 \\
C_2 &= x_1 \lor \neg x_2 \lor x_3 \\
C_3 &= \neg x_1 \lor \neg x_2 \lor \neg x_3
\end{aligned}
\]

### 3-satisfiability reduces to subset sum

**Lemma.** $\Phi$ is satisfiable iff there exists a subset that sums to $W$.

**Pf.** $\Rightarrow$ Suppose $\Phi$ is satisfiable.

- Choose integers corresponding to each *true* literal.
- Since $\Phi$ is satisfiable, each $C_j$ digit sums to at least 1 from $x_i$ rows.
- Choose dummy integers to make clause digits sum to 4.

![Subset-Sum instance diagram]
3-satisfiability reduces to subset sum

**Lemma.** \( \Phi \) is satisfiable iff there exists a subset that sums to \( W \).

**Pf.** \( \Leftarrow \) Suppose there is a subset that sums to \( W \).

- Digit \( x_i \) forces subset to select either row \( x_i \) or \( \neg x_i \) (but not both).
- Digit \( C_j \) forces subset to select at least one literal in clause.
- Assign \( x_i = true \) iff row \( x_i \) selected. 

\[
\begin{align*}
C_1 &= \neg x_1 \vee x_2 \vee x_3 \\
C_2 &= x_1 \vee \neg x_2 \vee x_3 \\
C_3 &= \neg x_1 \vee \neg x_2 \vee \neg x_3
\end{align*}
\]

3-SAT instance

\[
\begin{array}{lllllll}
\text{W} & 1 & 1 & 1 & 4 & 4 & 4 \\
& 111,444 & \\
\hline
\text{SUBSET-SUM instance}
\end{array}
\]
**Partition**

**SUBSET-SUM.** Given natural numbers \(w_1, \ldots, w_n\) and an integer \(W\), is there a subset that adds up to exactly \(W\)?

**PARTITION.** Given natural numbers \(v_1, \ldots, v_m\), can they be partitioned into two subsets that add up to the same value \(\frac{1}{2} \sum v_i\)?

**Theorem.** \(\text{SUBSET-SUM} \leq_p \text{PARTITION.}\)

**Pf.** Let \(W, w_1, \ldots, w_n\) be an instance of \(\text{SUBSET-SUM}.\)

- Create instance of \(\text{PARTITION}\) with \(m = n + 2\) elements.
  - \(v_1 = w_1, v_2 = w_2, \ldots, v_n = w_n, v_{n+1} = 2 \sum w_i - W, v_{n+2} = \sum w_i + W\)
- Lemma: there exists a subset that sums to \(W\) iff there exists a partition since elements \(v_{n+1}\) and \(v_{n+2}\) cannot be in the same partition.

<table>
<thead>
<tr>
<th>(v_{n+1} = 2 \sum w_i - W)</th>
<th>(W)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>subset A</strong></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(v_{n+2} = \sum w_i + W)</th>
<th>(\sum w_i - W)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>subset B</strong></td>
<td></td>
</tr>
</tbody>
</table>

\(1 + 3 = 4\)
Polynomial-time reductions

3-SAT

constraint satisfaction

INDEPENDENT-SET

DIR-HAM-CYCLE

GRAPH-3-COLOR

SUBSET-SUM

3-SAT poly-time reduces to INDEPENDENT-SET

VERTEX-COVER

HAM-CYCLE

PLANAR-3-COLOR

SCHEDULING

packing and covering

sequencing

partitioning

numerical
Karp's 21 NP-complete problems

![Diagram of Karp's 21 NP-complete problems]

Dick Karp (1972)
1985 Turing Award

**Figure 1 - Complete Problems**