Chapter 8

NP and Computational Intractability
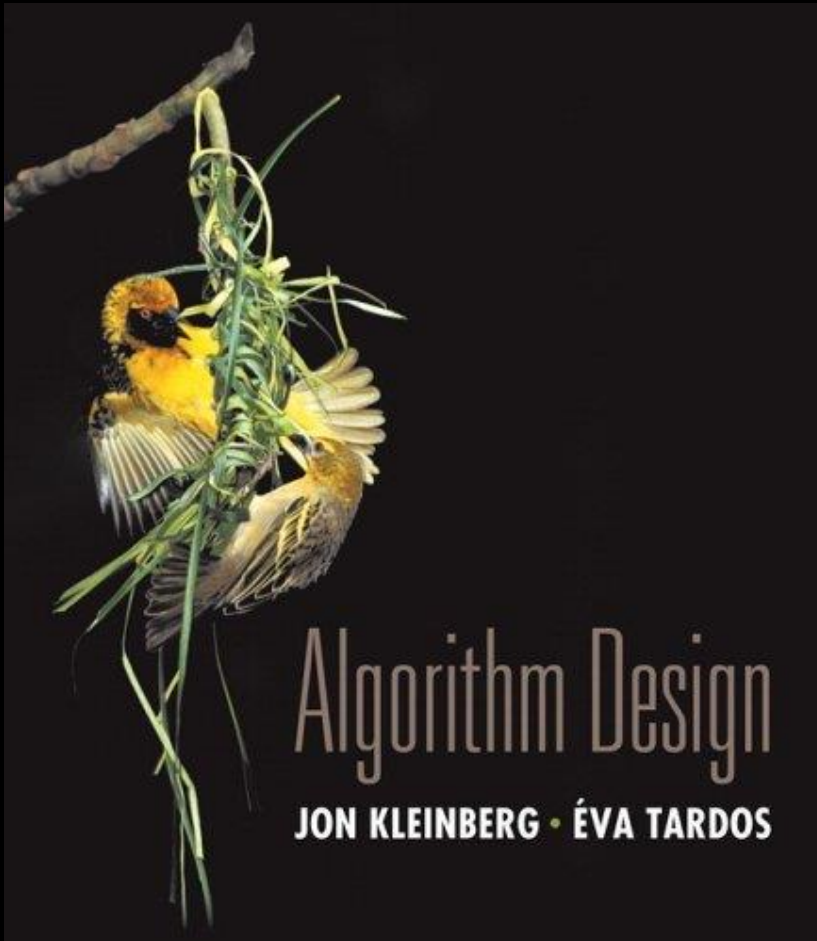
Algorithm Design

JON KLEINBERG · ÉVA TARDOS

8.3 Definition of NP

Decision Problems

Decision problem.

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

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

↑
length of s

PRIMES: X = { 2, 3, 5, 7, 11, 13, 17, 23, 29, 31, 37, ... }
Algorithm. [Agrawal-Kayal-Saxena, 2002] $p(|s|) = |s|^8$.

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
RELPRIME	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
LSOLVE	Is there a vector x that satisfies Ax = b?	Gauss-Edmonds elimination	$\begin{vmatrix} 0 & 1 & 1 \\ 2 & 4 & -2 \\ 0 & 3 & 15 \end{vmatrix}, \begin{vmatrix} 4 \\ 2 \\ 36 \end{vmatrix}$	$\begin{vmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{vmatrix}, \begin{vmatrix} 1 \\ 1 \\ 1 \end{vmatrix}$

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"

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

$C(s, t)$ is a poly-time algorithm and $|t| \leq p(|s|)$ 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. Note that such a certificate exists iff s is composite. Moreover $|t| \leq |s|$.

Certifier.

```
boolean C(s, t) {
    if (t ≤ 1 or t ≥ s)
        return false
    else if (s is a multiple of t)
        return true
    else
        return false
}
```

Instance. s = 437,669.
Certificate. t = 541 or 809. ⟵ 437,669 = 541 × 809

Conclusion. COMPOSITES is in NP.

Certifiers and Certificates: 3-Satisfiability

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.

Ex.

$$\left(\overline{x_1} \vee x_2 \vee x_3 \right) \wedge \left(x_1 \vee \overline{x_2} \vee x_3 \right) \wedge \left(x_1 \vee x_2 \vee x_4 \right) \wedge \left(\overline{x_1} \vee \overline{x_3} \vee \overline{x_4} \right)$$

instance s

$$x_1 = 1, \ x_2 = 1, \ x_3 = 0, \ x_4 = 1$$

certificate t

Conclusion. SAT is in NP.

Certifiers and Certificates: Hamiltonian Cycle

HAM-CYCLE. Given an undirected graph G = (V, E), does there exist a simple cycle C 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 in the permutation.

Conclusion. HAM-CYCLE is in NP.

instance s

certificate t
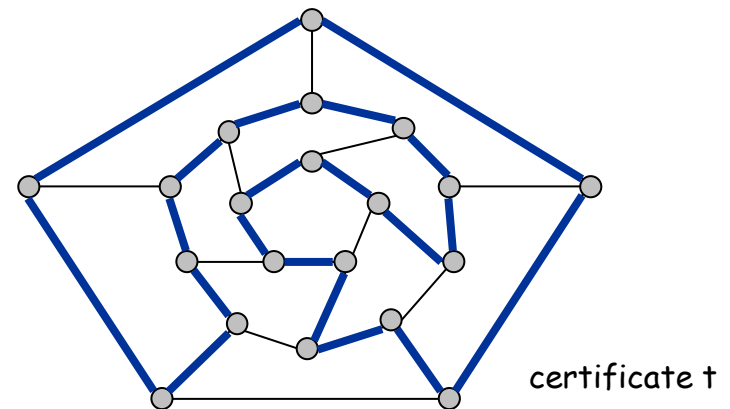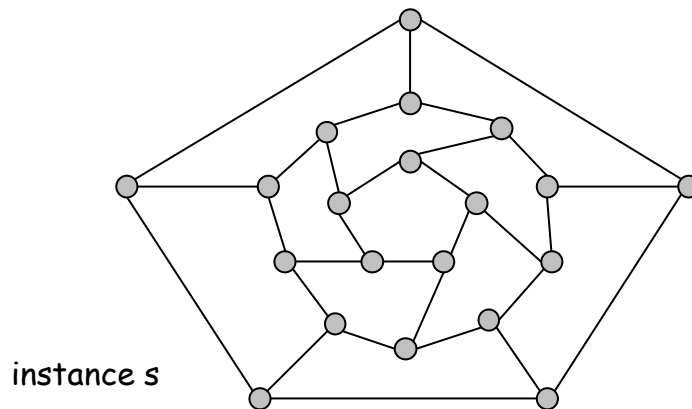
P, NP, EXP

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

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 = ε, certifier C(s, t) = A(s). ∎

Claim. NP $\subseteq$ EXP.
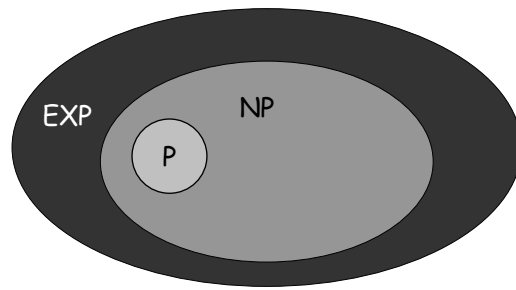Pf. Consider any problem X in NP.
- By definition, there exists a poly-time certifier C(s, t) for X.
- To solve input s, run C(s, t) on all strings t with $|t| \leq p(|s|)$.
- Return `yes`, if C(s, t) returns `yes` for any of these. ∎
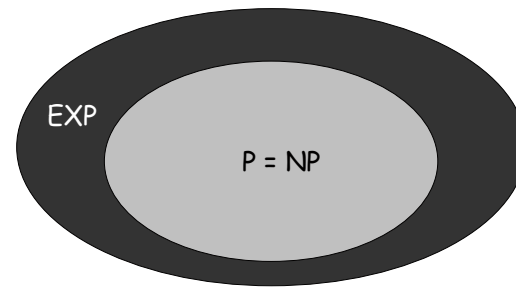
The Main Question: P Versus NP

Does P = NP? [Cook 1971, Edmonds, Levin, Yablonski, Gödel]

- Is the decision problem as easy as the certification problem?
- Clay $1 million prize.

If P ≠ NP   If P = NP

would break RSA cryptography
(and potentially collapse economy)

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

Consensus opinion on P = NP? Probably no.

8.4 NP-Completeness

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 is an NP-complete problem. Then Y is solvable in poly-time iff P = NP.

Pf. $\Leftarrow$ If P = NP then Y can be solved in poly-time since Y is in NP.

Pf. $\Rightarrow$ Suppose Y can be solved in poly-time.
- Let X be any problem in NP. Since $X \leq_p Y$, we can solve X in poly-time. This implies NP $\subseteq$ P.
- We already know P $\subseteq$ NP. Thus P = NP. ∎

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

Establishing NP-Completeness

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

Recipe to establish NP-completeness of problem Y.
- Step 1. Show that Y is in NP.
- Step 2. Choose an NP-complete problem X.
- Step 3. Prove that $X \leq_p Y$.

Justification. If X is an NP-complete problem, and Y is a problem in NP with the property that $X \leq_P Y$ then Y is NP-complete.

Pf. Let W be any problem in NP. Then $W \leq_P X \leq_P Y$.
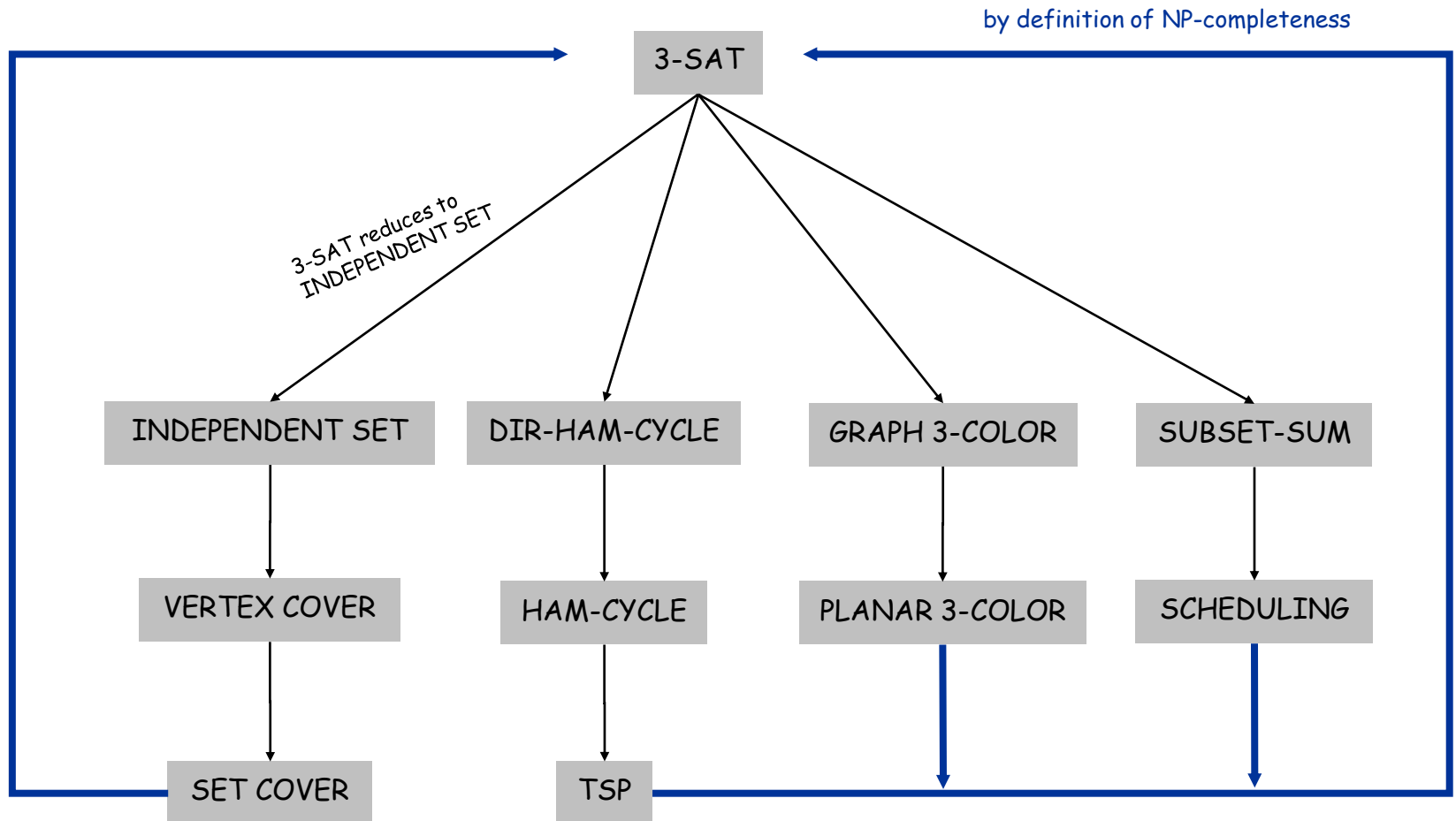- By transitivity, $W \leq_P Y$.
- Hence Y is NP-complete. ∎

    ↑   ↑

by definition of   by assumption
NP-complete

Theorem. 3-SAT is NP-complete (Cook 1971).

NP-Completeness

Observation. All problems below are NP-complete and polynomial reduce to one another!

Some NP-Complete Problems

Six basic genres of NP-complete problems and paradigmatic examples.

- Packing problems: SET-PACKING, INDEPENDENT SET.
- Covering problems: SET-COVER, VERTEX-COVER.
- Constraint satisfaction problems: SAT, 3-SAT.
- Sequencing problems: HAMILTONIAN-CYCLE, TSP.
- Partitioning problems: 3D-MATCHING 3-COLOR.
- Numerical problems: SUBSET-SUM, KNAPSACK.

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

Notable exceptions. Factoring, graph isomorphism, Nash equilibrium.

Extent and Impact of NP-Completeness

Extent of NP-completeness. [Papadimitriou 1995]

- Prime intellectual export of CS to other disciplines.
- 6,000 citations per year (title, abstract, keywords).
  - more than "compiler", "operating system", "database"
- Broad applicability and classification power.
- "Captures vast domains of computational, scientific, mathematical endeavors, and seems to roughly delimit what mathematicians and scientists had been aspiring to compute feasibly."

NP-completeness can guide scientific inquiry.

- 1926: Ising introduces simple model for phase transitions.
- 1944: Onsager solves 2D case in tour de force.
- 19xx: Feynman and other top minds seek 3D solution.
- 2000: Istrail proves 3D problem NP-complete.

More Hard Computational Problems

Aerospace engineering: optimal mesh partitioning for finite elements.

Biology: protein folding.

Chemical engineering: heat exchanger network synthesis.

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: find minimum risk portfolio of given return.

Game theory: find Nash equilibrium that maximizes social welfare.

Genomics: phylogeny reconstruction.

Mechanical engineering: structure of turbulence in sheared flows.

Medicine: reconstructing 3-D shape from biplane angiocardiogram.

Operations research: optimal resource allocation.

Physics: partition function of 3-D Ising model in statistical mechanics.

Politics: Shapley-Shubik voting power.

Pop culture: Minesweeper consistency.

Statistics: optimal experimental design.

8.5 Sequencing Problems

Basic genres.

- Packing problems: SET-PACKING, INDEPENDENT SET.
- Covering problems: SET-COVER, VERTEX-COVER.
- Constraint satisfaction problems: SAT, 3-SAT.
- Sequencing problems: HAMILTONIAN-CYCLE, TSP.
- Partitioning problems: 3D-MATCHING, 3-COLOR.
- Numerical problems: SUBSET-SUM, KNAPSACK.

Hamiltonian Cycle
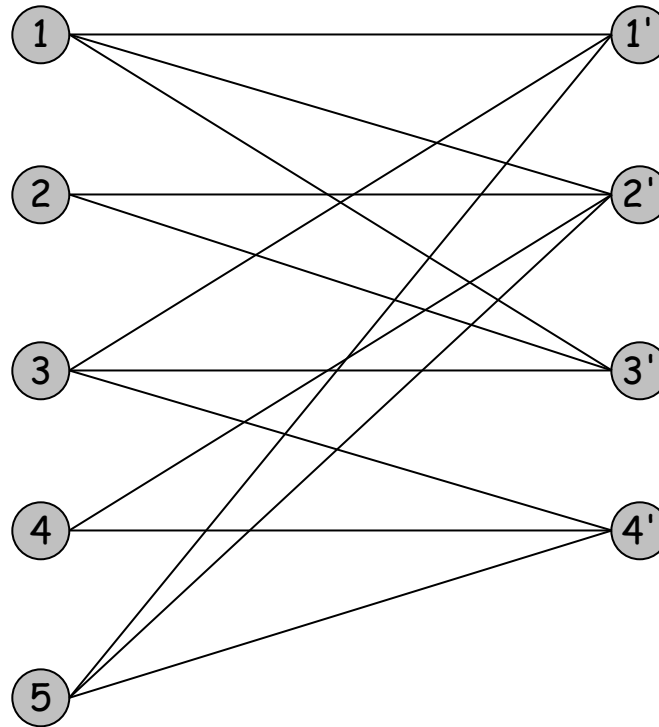
HAM-CYCLE: given an undirected graph G = (V, E), does there exist a simple cycle Γ that contains every node in V.
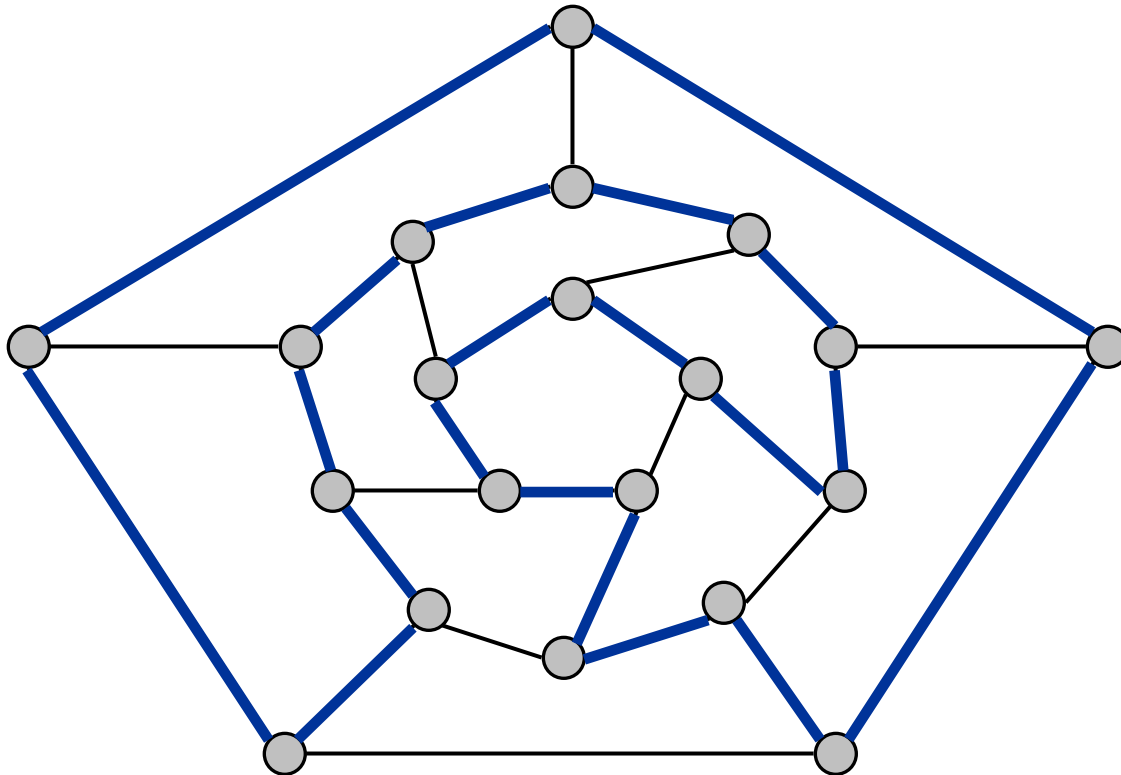
YES: vertices and faces of a dodecahedron.

Hamiltonian Cycle

HAM-CYCLE: given an undirected graph G = (V, E), does there exist a simple cycle Γ that contains every node in V.
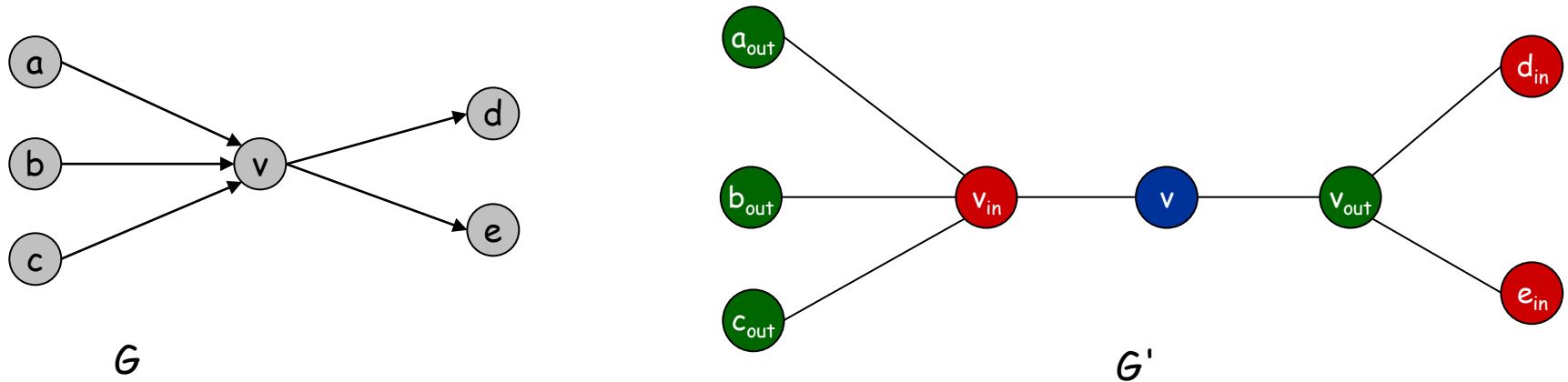

NO: bipartite graph with odd number of nodes.

Directed Hamiltonian Cycle

DIR-HAM-CYCLE: given a digraph $G = (V, E)$, does there exists a simple directed cycle Γ that contains every node in V?

Claim. DIR-HAM-CYCLE $\leq_P$ HAM-CYCLE.

Pf. Given a directed graph $G = (V, E)$, construct an undirected graph G' with 3n nodes.

G

G'

Directed Hamiltonian Cycle

Claim. G has a Hamiltonian cycle iff G' does.

Pf. $\Rightarrow$

- Suppose G has a directed Hamiltonian cycle Γ.
- Then G' has an undirected Hamiltonian cycle (same order).

Pf. $\Leftarrow$

- Suppose G' has an undirected Hamiltonian cycle Γ'.
- Γ' must visit nodes in G' using one of following two orders:

  …, B, G, R, B, G, R, B, G, R, B, …

  …, B, R, G, B, R, G, B, R, G, B, …

- Blue nodes in Γ' make up directed Hamiltonian cycle Γ in G, or reverse of one. ∎

3-SAT Reduces to Directed Hamiltonian Cycle

Claim. 3-SAT $\leq_P$ DIR-HAM-CYCLE.

Pf. Given an instance Φ of 3-SAT, we construct an instance of DIR-HAM-CYCLE that has a Hamiltonian cycle iff Φ is satisfiable.

Construction. First, create graph that has 2^n Hamiltonian cycles which correspond in a natural way to 2^n possible truth assignments.

3-SAT Reduces to Directed Hamiltonian Cycle

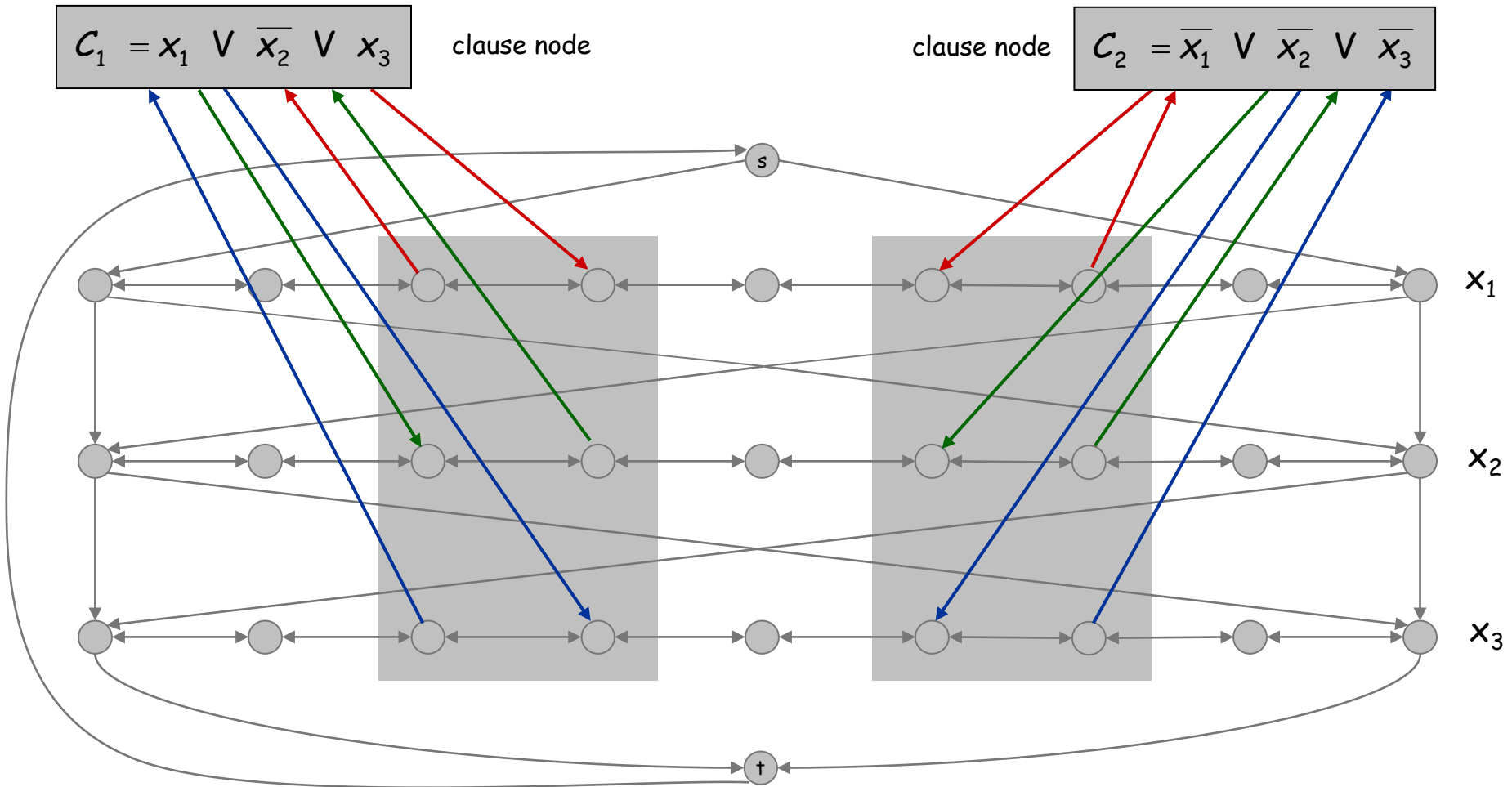Construction. Given 3-SAT instance Φ with n variables x_i and k clauses.

- Construct G to have 2^n Hamiltonian cycles.
- Intuition: traverse path i from left to right $\Leftrightarrow$ set variable x_i = 1.
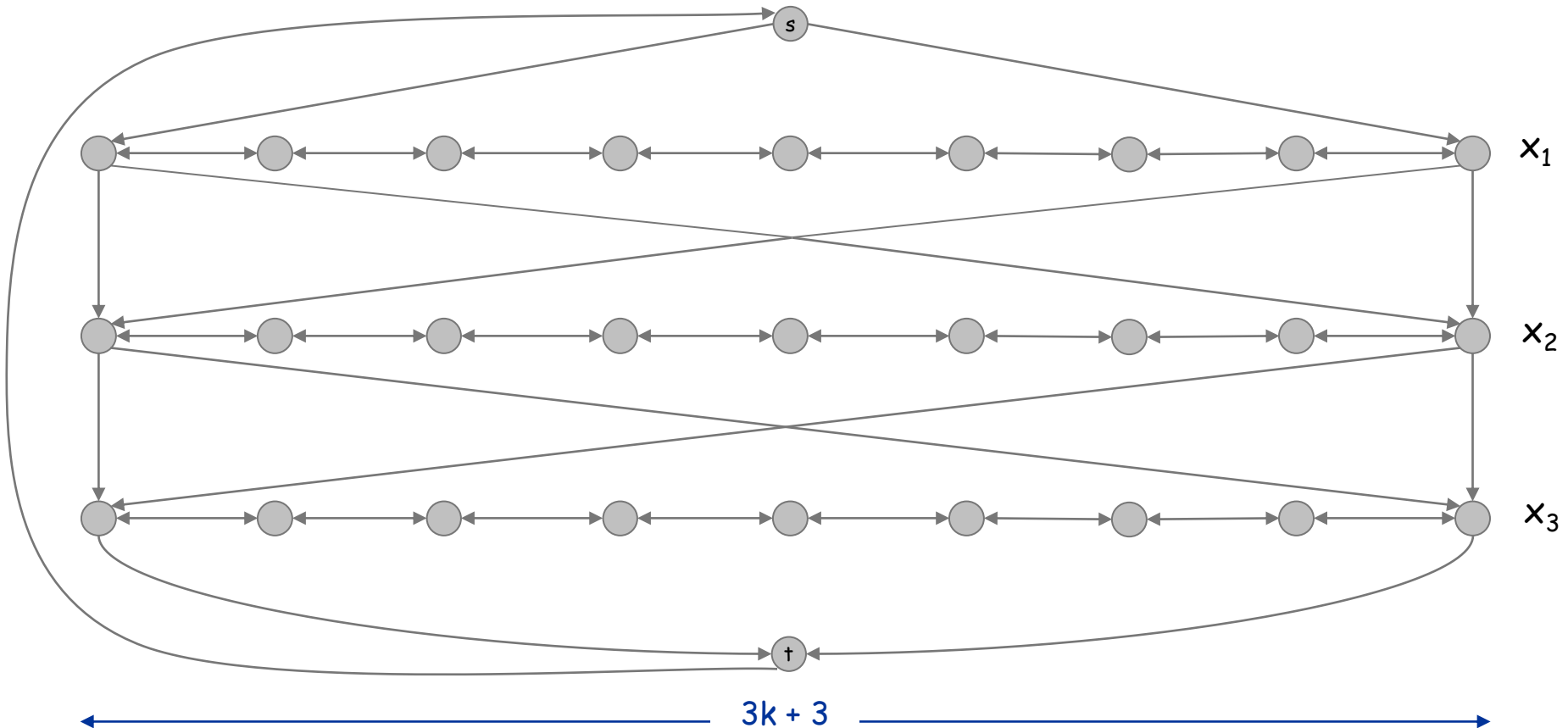
3-SAT Reduces to Directed Hamiltonian Cycle

Construction. Given 3-SAT instance Φ with n variables x_i and k clauses.

◻ For each clause: add a node and 6 edges.

$C_1 = x_1 \lor \overline{x_2} \lor x_3$ clause node   clause node $C_2 = \overline{x_1} \lor \overline{x_2} \lor \overline{x_3}$

x_1

x_2

x_3

3-SAT Reduces to Directed Hamiltonian Cycle

Claim. Φ is satisfiable iff G has a Hamiltonian cycle.

Pf. $\Rightarrow$

- Suppose 3-SAT instance has satisfying assignment x^*.
- Then, define Hamiltonian cycle in G as follows:
  - if $x^*_i = 1$, traverse row i from left to right
  - if $x^*_i = 0$, traverse row i from right to left
  - for each clause C_j , there will be at least one row i in which we are going in "correct" direction to splice node C_j into tour

3-SAT Reduces to Directed Hamiltonian Cycle

Claim. Φ is satisfiable iff G has a Hamiltonian cycle.

Pf. $\Leftarrow$

- Suppose G has a Hamiltonian cycle Γ.
- If Γ enters clause node C_j , it must depart on mate edge.
  - thus, nodes immediately before and after C_j are connected by an edge e in G
  - removing C_j from cycle, and replacing it with edge e yields Hamiltonian cycle on G - { C_j }
- Continuing in this way, we are left with Hamiltonian cycle Γ' in G - { C_1 , C_2 , . . . , C_k }.
- Set $x^*_i = 1$ iff Γ' traverses row i left to right.
- Since Γ visits each clause node C_j , at least one of the paths is traversed in "correct" direction, and each clause is satisfied.   ▪
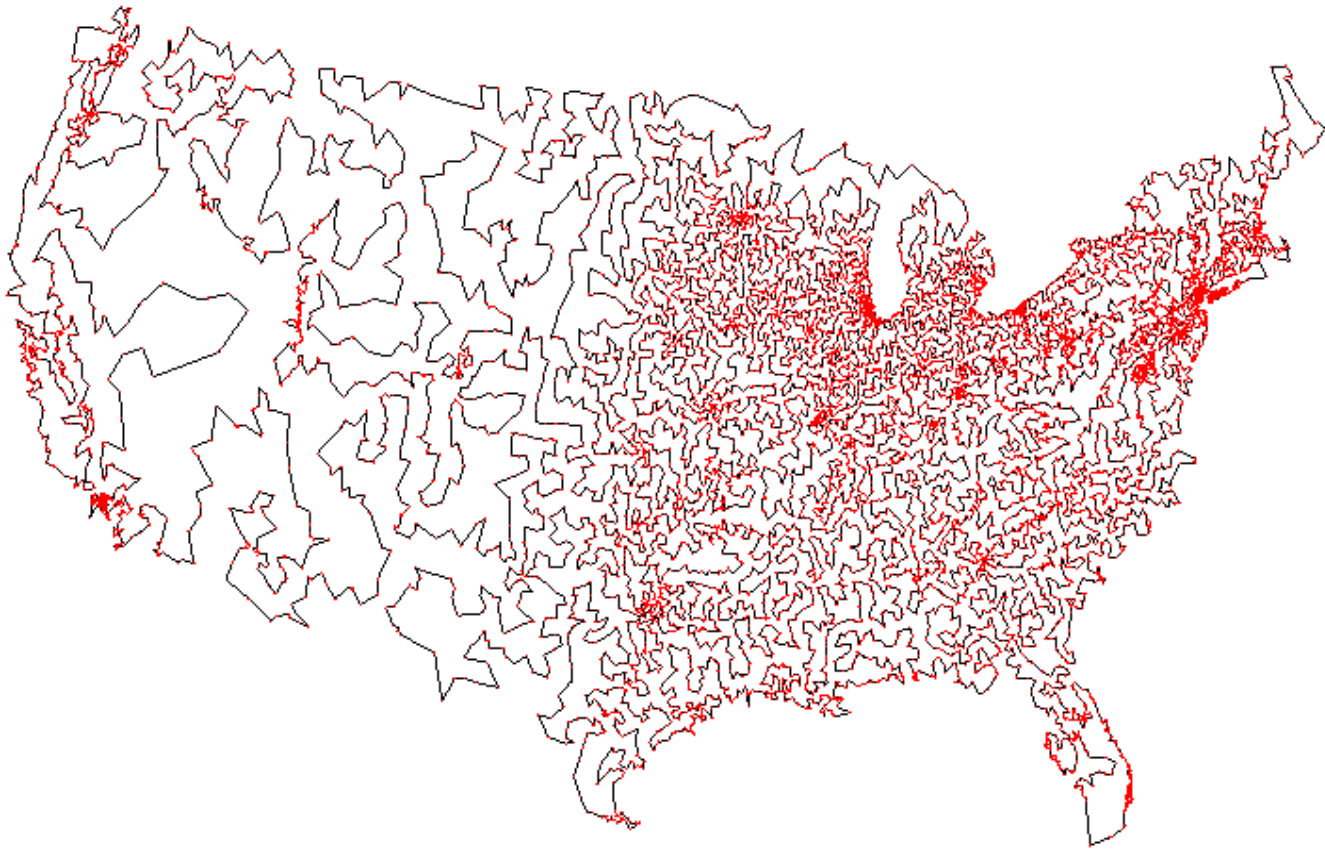
Traveling Salesperson Problem

TSP. Given a set of n cities and a pairwise distance function d(u, v), is there a tour of length ≤ D?
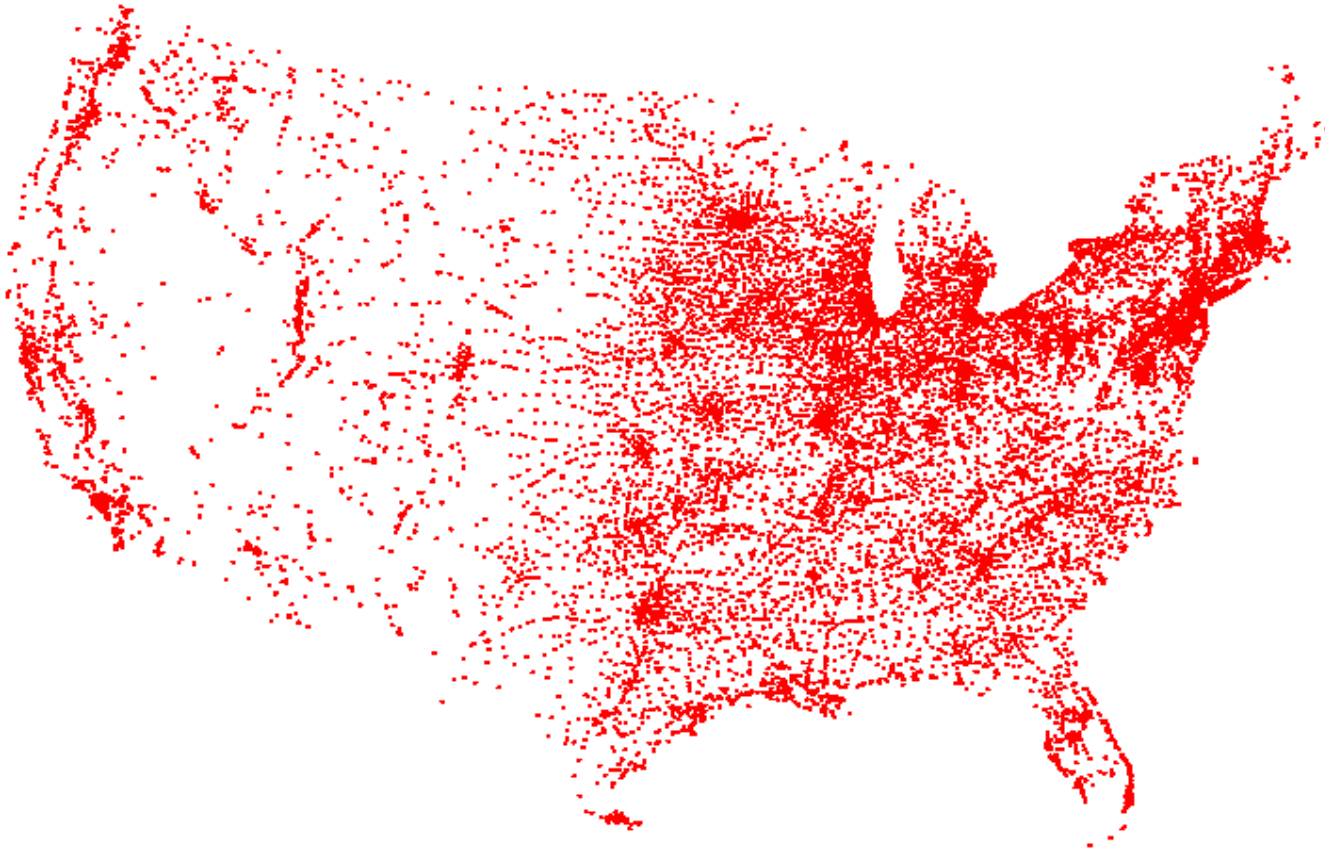
All 13,509 cities in US with a population of at least 500
Reference: http://www.tsp.gatech.edu

Traveling Salesperson Problem

TSP. Given a set of n cities and a pairwise distance function d(u, v), is there a tour of length ≤ D?

Optimal TSP tour
Reference: http://www.tsp.gatech.edu

Traveling Salesperson Problem

TSP. Given a set of n cities and a pairwise distance function d(u, v), is there a tour of length $\leq$ D?

HAM-CYCLE: given a graph G = (V, E), does there exists a simple cycle that contains every node in V?

Claim. HAM-CYCLE $\leq_P$ TSP.

Pf.

- Given instance G = (V, E) of HAM-CYCLE, create n cities with distance function

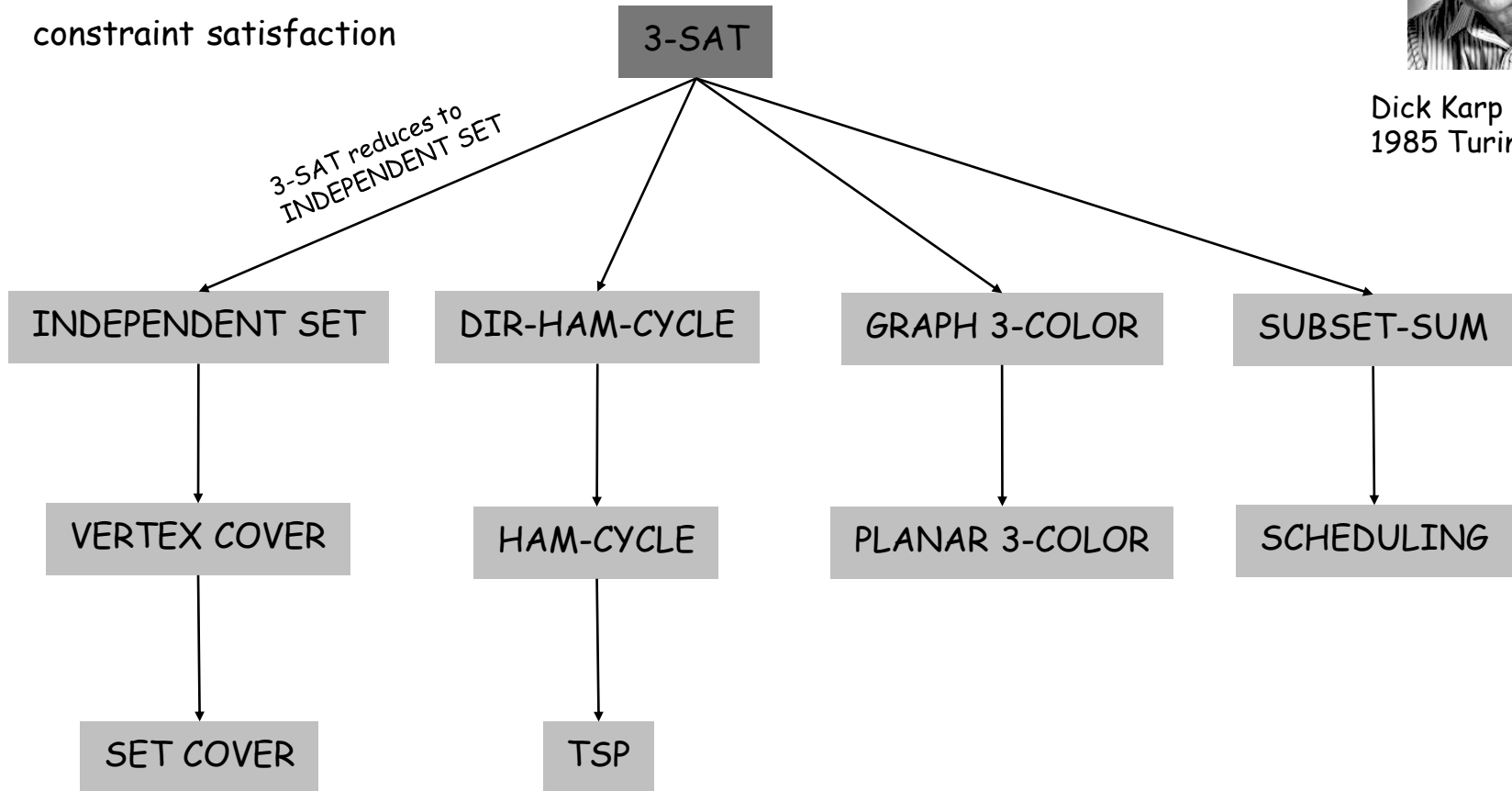$$d(u, v) = \begin{cases} 1 & \text{if}\,(u,\ v)\ \in\ E \\ 2 & \text{if}\,(u,\ v)\ \notin\ E \end{cases}$$

- TSP instance has tour of length $\leq$ n iff G is Hamiltonian. ∎

Remark. TSP instance in reduction satisfies Δ-inequality.

Polynomial-Time Reductions

constraint satisfaction

3-SAT

Dick Karp (1972)
1985 Turing Award

3-SAT reduces to
INDEPENDENT SET

INDEPENDENT SET

DIR-HAM-CYCLE

GRAPH 3-COLOR

SUBSET-SUM

VERTEX COVER

HAM-CYCLE

PLANAR 3-COLOR

SCHEDULING

SET COVER

TSP

packing and covering

sequencing

partitioning

numerical