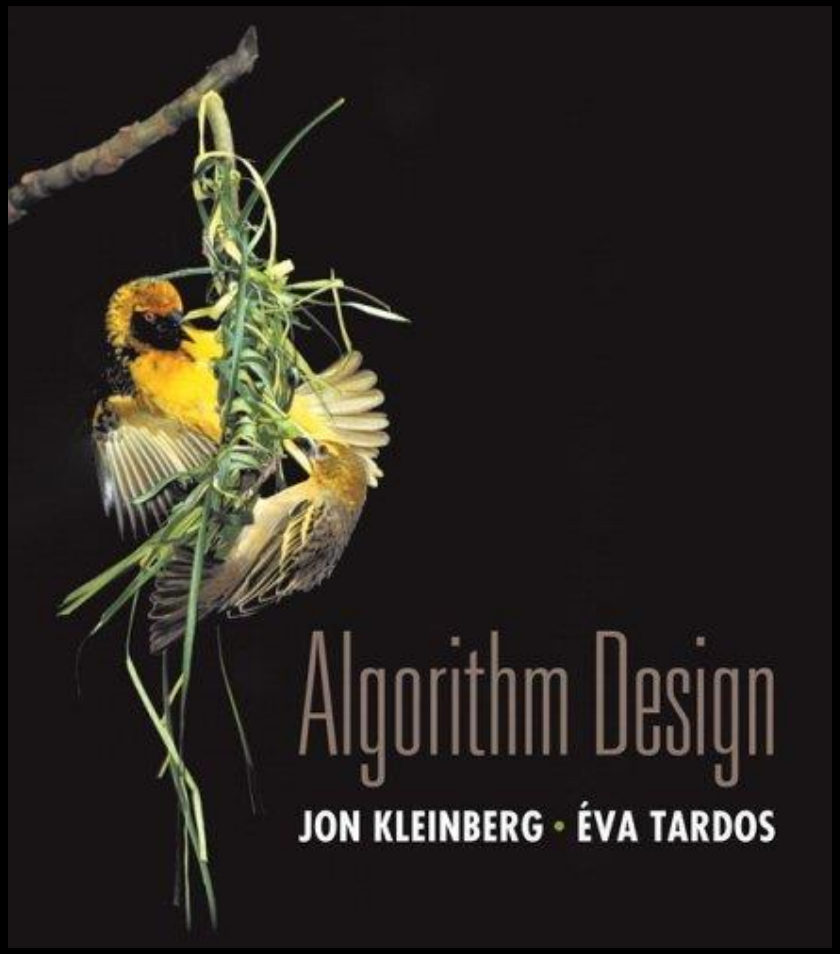


Chapter 8

NP and Computational Intractability

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(·) is some polynomial.

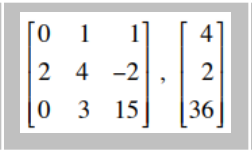
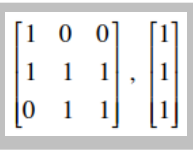
length of s

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		

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

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

Conclusion. SAT is in NP.

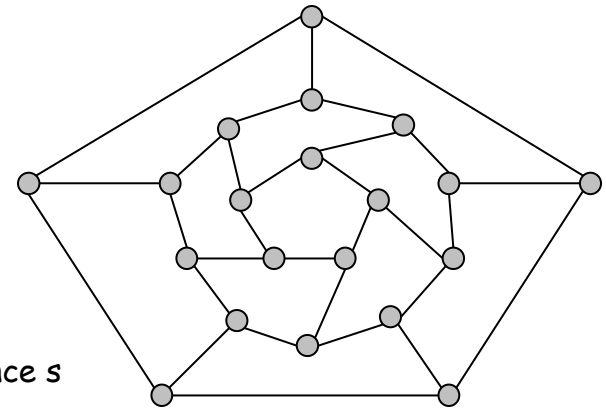
Certifiers and Certificates: Hamiltonian Cycle

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

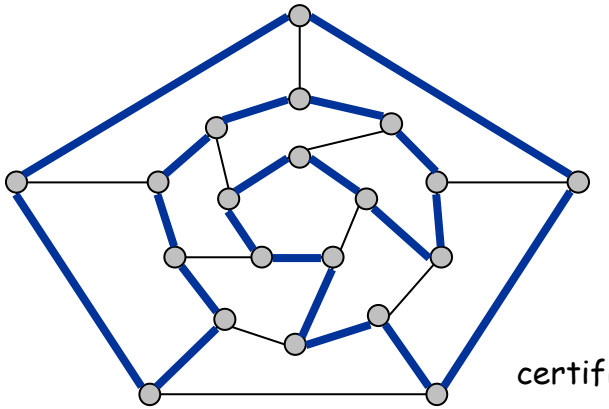
Certificate. A permutation of the n nodes.

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

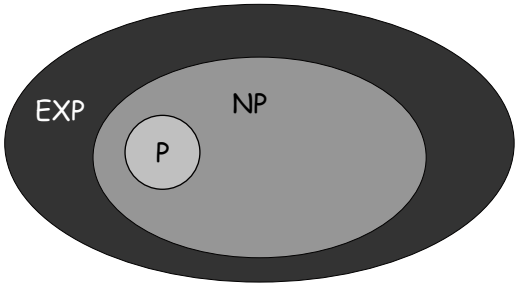
Claim. $P \subseteq NP$.
Pf. Consider any problem X in P.
- By definition, there exists a poly-time algorithm $A(s)$ that solves X.
- Certificate: $t = \varepsilon$, certifier $C(s, t) = A(s)$. ▪

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. ▪

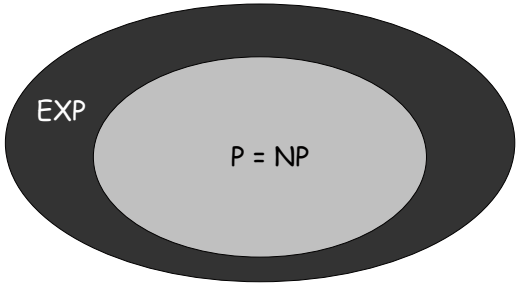
The Main Question: P Versus NP

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

Polynomial Transformation

Def. Problem X polynomial reduces (Cook) 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 transforms (Karp) 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?

↑ 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 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.

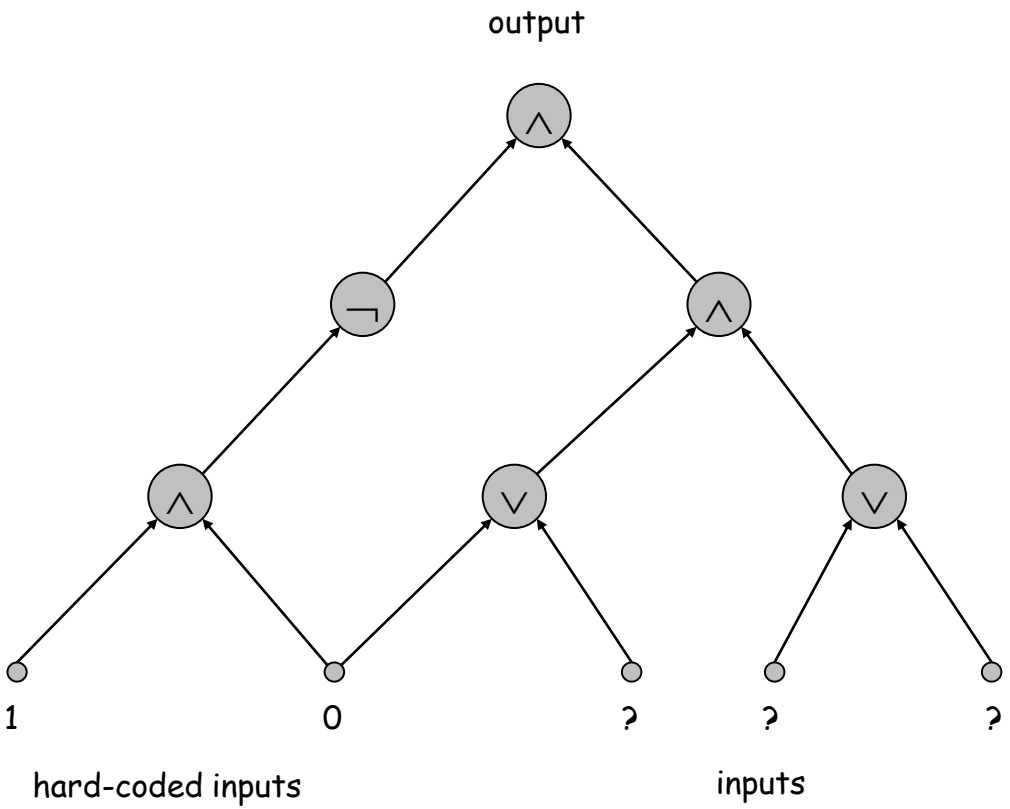
- Let X be any problem in NP. Since $X \leq_p Y$, we can solve X in poly-time. This implies NP $\subseteq$ P.
- We already know P $\subseteq$ NP. Thus P = NP. ▪

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

Circuit Satisfiability

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

yes: 1 0 1



The "First" NP-Complete Problem

Theorem. CIRCUIT-SAT is NP-complete. [Cook 1971, Levin 1973]

Pf. (sketch)

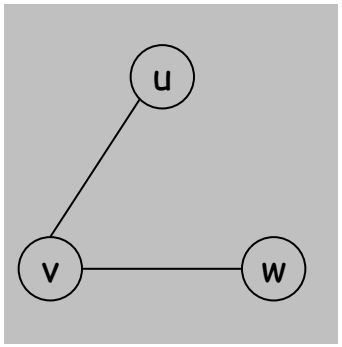
- Any algorithm that takes a fixed number of bits n as input and produces a yes/no answer can be represented by such a circuit. Moreover, if algorithm takes poly-time, then circuit is of poly-size.

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

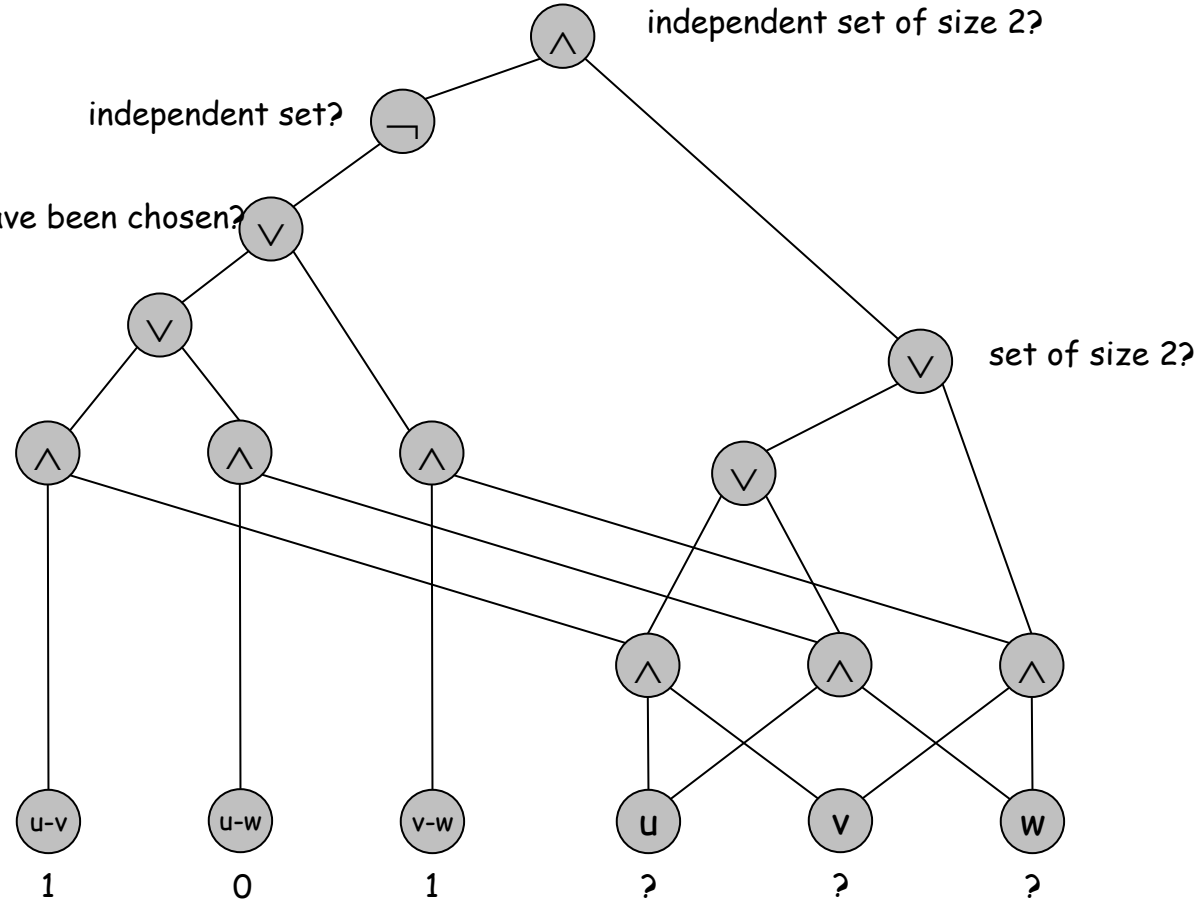
- Consider some problem X in NP. It has a poly-time certifier $C(s, t)$. To determine whether s is in X, need to know if there exists a certificate t of length $p(|s|)$ such that $C(s, t) =$ `yes`.
- View $C(s, t)$ as an algorithm on $|s| + p(|s|)$ bits (input s, certificate t) and convert it into a poly-size circuit K.
  - first $|s|$ bits are hard-coded with s
  - remaining $p(|s|)$ bits represent 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 K outputs true iff graph G has an independent set of size 2.



G = (V, E), n = 3



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. ▪

$W \leq_P X$: by definition of NP-complete

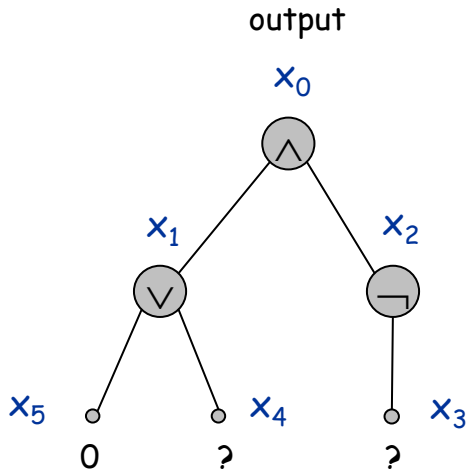
$X \leq_P Y$: by assumption

3-SAT is NP-Complete

Theorem. 3-SAT is NP-complete.

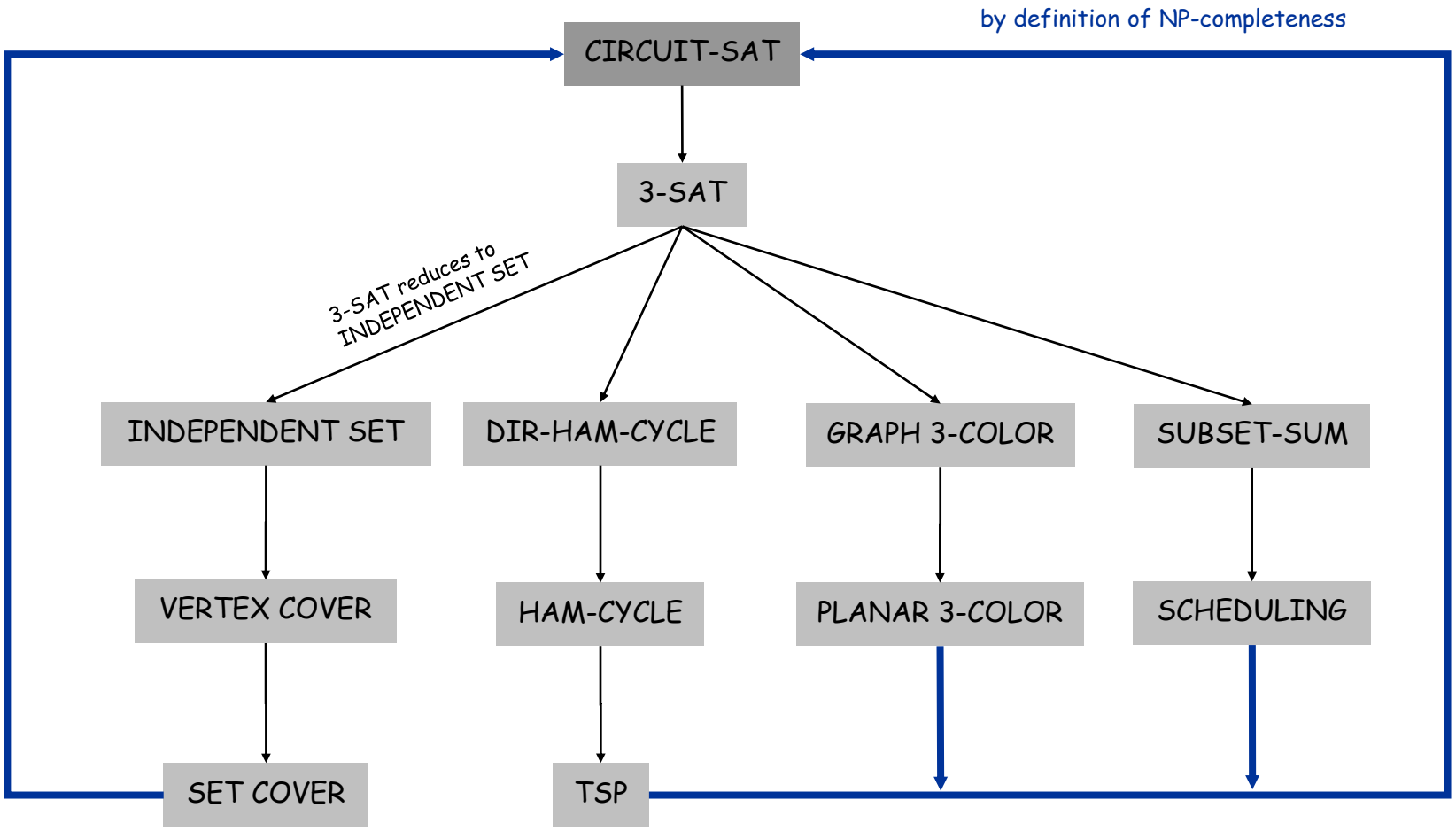
Pf. Suffices to show that CIRCUIT-SAT $\leq_P$ 3-SAT since 3-SAT is in NP.

- Let K be any circuit.
- Create a 3-SAT variable x_i for each circuit element i.
- Make circuit compute correct values at each node:
  - $x_2 = \neg x_3 \Rightarrow$ add 2 clauses: $x_2 \vee x_3$, $\neg x_2 \vee \neg x_3$
  - $x_1 = x_4 \vee x_5 \Rightarrow$ add 3 clauses: $(x_1 \vee \neg x_4)$, $(x_1 \vee \neg x_5)$, $(\neg x_1 \vee x_4 \vee x_5)$
  - $x_0 = x_1 \wedge x_2 \Rightarrow$ add 3 clauses: $(\neg x_0 \vee x_1)$, $(\neg x_0 \vee x_2)$, $(x_0 \vee \neg x_1 \vee \neg x_2)$
- Hard-coded input values and output value.
  - $x_5 = 0 \Rightarrow$ add 1 clause: $\neg x_5$
  - $x_0 = 1 \Rightarrow$ add 1 clause: x_0
- Final step: turn clauses of length < 3 into clauses of length exactly 3. ▪



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.

8.9 co-NP and the Asymmetry of NP

Asymmetry of NP

Asymmetry of NP. We only need to have short proofs of `yes` instances.

Ex 1. SAT vs. TAUTOLOGY.

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

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

- Can prove a graph is Hamiltonian by giving such a Hamiltonian cycle.
- How could we prove that a graph is not Hamiltonian?

Remark. SAT is NP-complete and SAT $\equiv_P$ TAUTOLOGY, but how do we classify TAUTOLOGY?

↑

not even known to be in NP

NP and co-NP

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

Ex. SAT, HAM-CYCLE, COMPOSITES.

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

Ex. $\overline{X}$ = { 0, 1, 4, 6, 8, 9, 10, 12, 14, 15, … }
X = { 2, 3, 5, 7, 11, 13, 17, 23, 29, … }

co-NP. Complements of decision problems in NP.

Ex. TAUTOLOGY, NO-HAM-CYCLE, PRIMES.

NP = co-NP ?

Fundamental question. Does NP = co-NP?

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

Theorem. If NP ≠ co-NP, then P ≠ NP.

Pf idea.

- P is closed under complementation.
- If P = NP, then NP is closed under complementation.
- In other words, NP = co-NP.
- This is the contrapositive of the theorem.

Good Characterizations

Good characterization. [Edmonds 1965] NP ∩ co-NP.

- If problem X is in both 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 $|N(S)| < |S|$.

Good Characterizations

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

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

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

- Mixed opinions.
- Many examples where problem found to have a non-trivial good characterization, but only years later discovered to be in P.
  - linear programming [Khachiyan, 1979]
  - primality testing [Agrawal-Kayal-Saxena, 2002]

Fact. Factoring is in $NP \cap co\text{-}NP$, but not known to be in P.

if poly-time algorithm for factoring, can break RSA cryptosystem

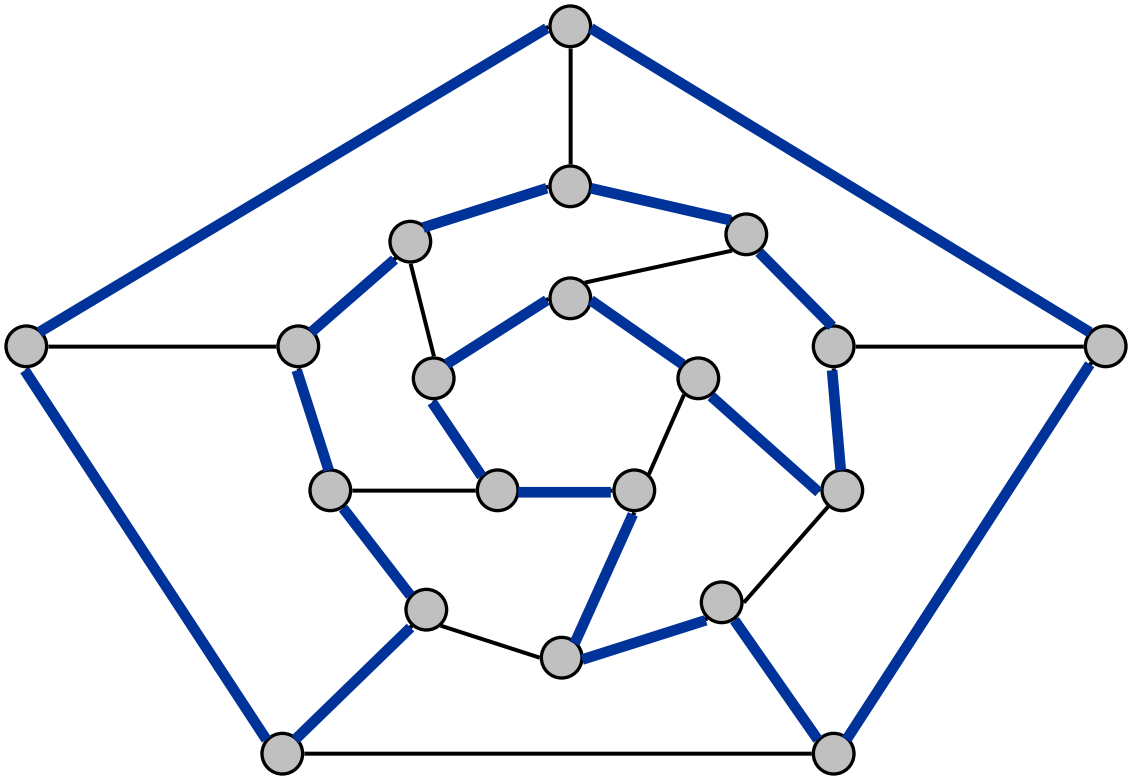
8.5 Sequencing Problems

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

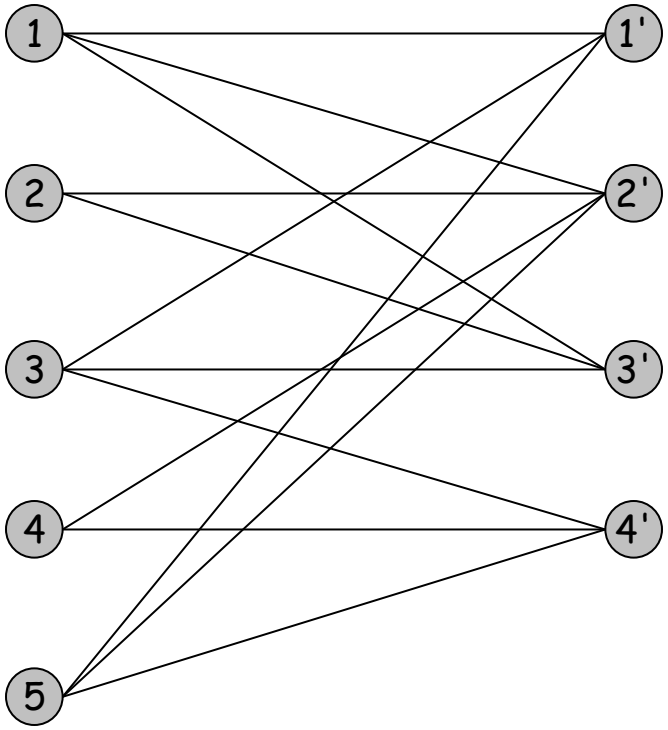
HAM-CYCLE: given an undirected graph $G = (V, E)$, does there exist a simple cycle Γ that contains every node in V.



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.



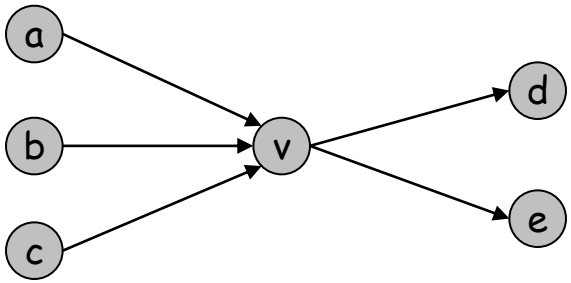
NO: bipartite graph with odd number of nodes.

Directed Hamiltonian Cycle

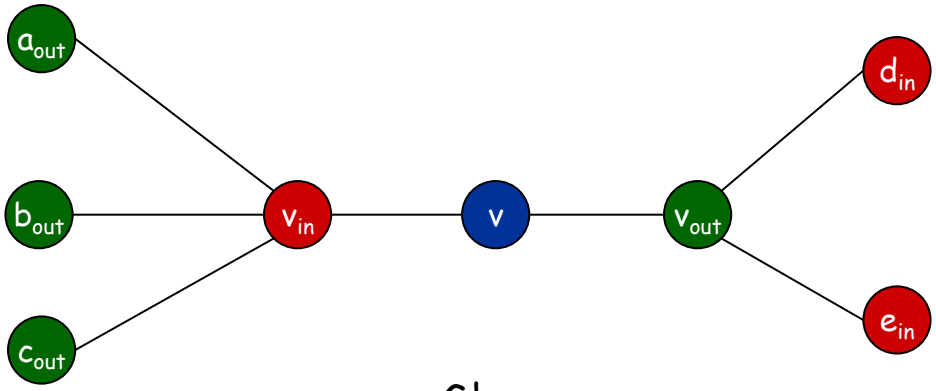
DIR-HAM-CYCLE: given a digraph $G = (V, E)$, does there exists a simple directed cycle Γ that contains every node in V?

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.

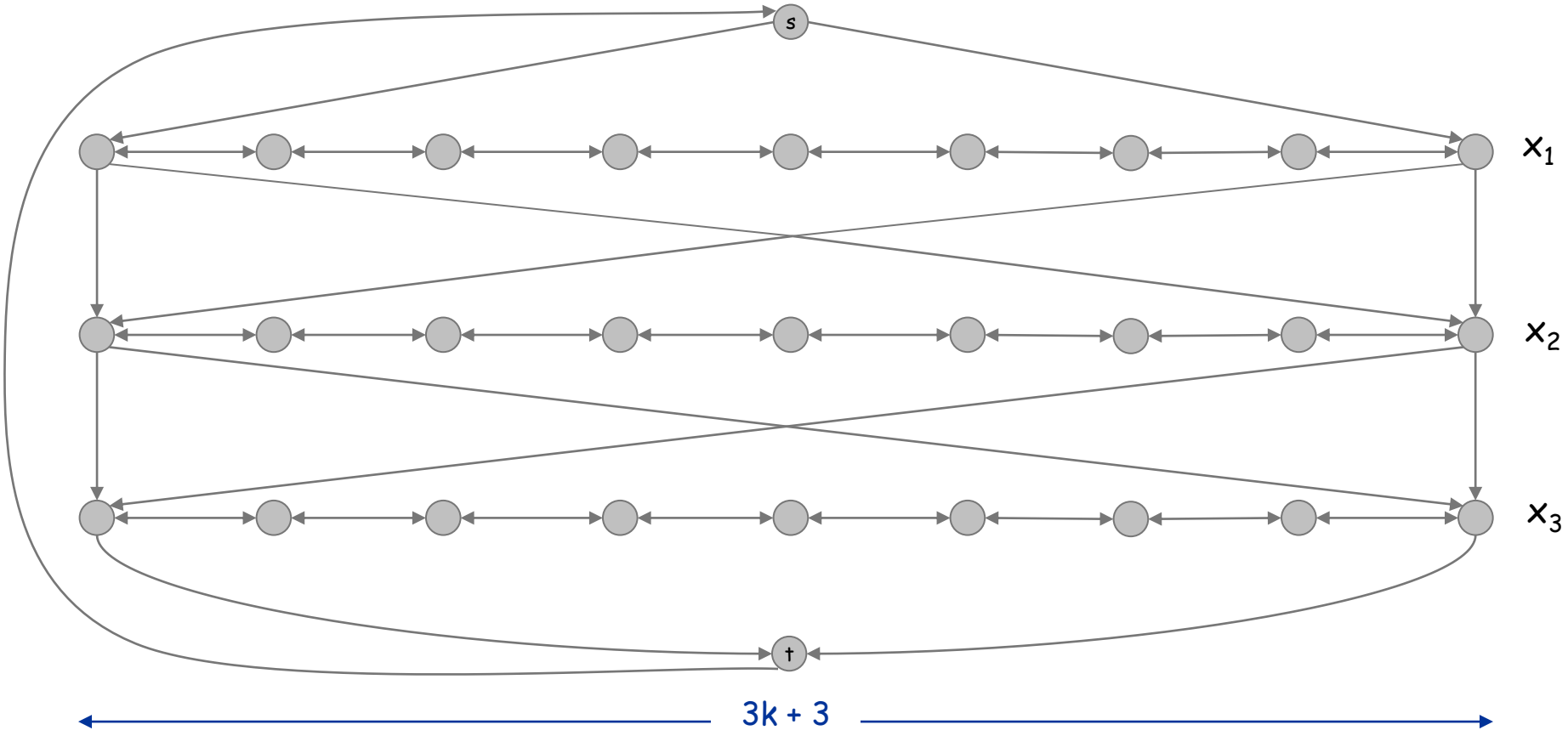
Pf. Given an instance Φ of 3-SAT, we construct an instance of DIR-HAM-CYCLE that has a Hamiltonian cycle iff Φ is satisfiable.

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.

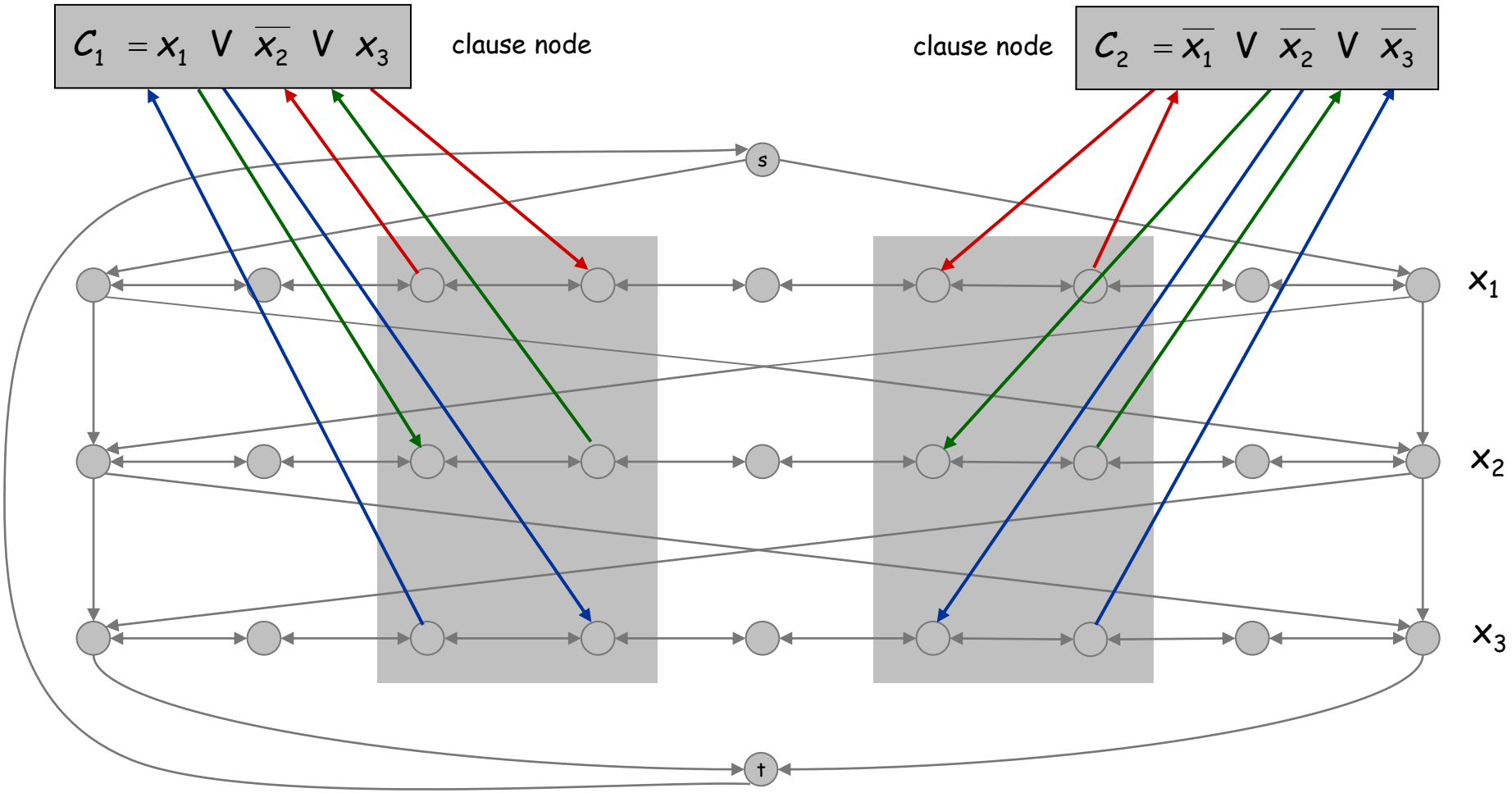
- Construct G to have 2^n Hamiltonian cycles.
- Intuition: traverse path i from left to right $\Leftrightarrow$ set variable $x_i = 1$.



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.



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 $G - \{ C_1, C_2, \ldots, C_k \}$.
- 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. ▪

Longest Path

SHORTEST-PATH. Given a digraph $G = (V, E)$, does there exists a simple path of length at most k edges?

LONGEST-PATH. Given a digraph $G = (V, E)$, does there exists a simple path of length at least k edges?

Claim. 3-SAT $\leq_P$ LONGEST-PATH.

Pf 1. Redo proof for DIR-HAM-CYCLE, ignoring back-edge from t to s.

Pf 2. Show HAM-CYCLE $\leq_P$ LONGEST-PATH.

The Longest Path †

Lyrics. Copyright © 1988 by Daniel J. Barrett.
Music. Sung to the tune of *The Longest Time* by Billy Joel.

Woh-oh-oh-oh, find the longest path!
Woh-oh-oh-oh, find the longest path!

If you said P is NP tonight,
There would still be papers left to write,
I have a weakness,
I'm addicted to completeness,
And I keep searching for the longest path.

The algorithm I would like to see
Is of polynomial degree,
But it's elusive:
Nobody has found conclusive
Evidence that we can find a longest path.

I have been hard working for so long.
I swear it's right, and he marks it wrong.
Some how I'll feel sorry when it's done:
GPA 2.1
Is more than I hope for.

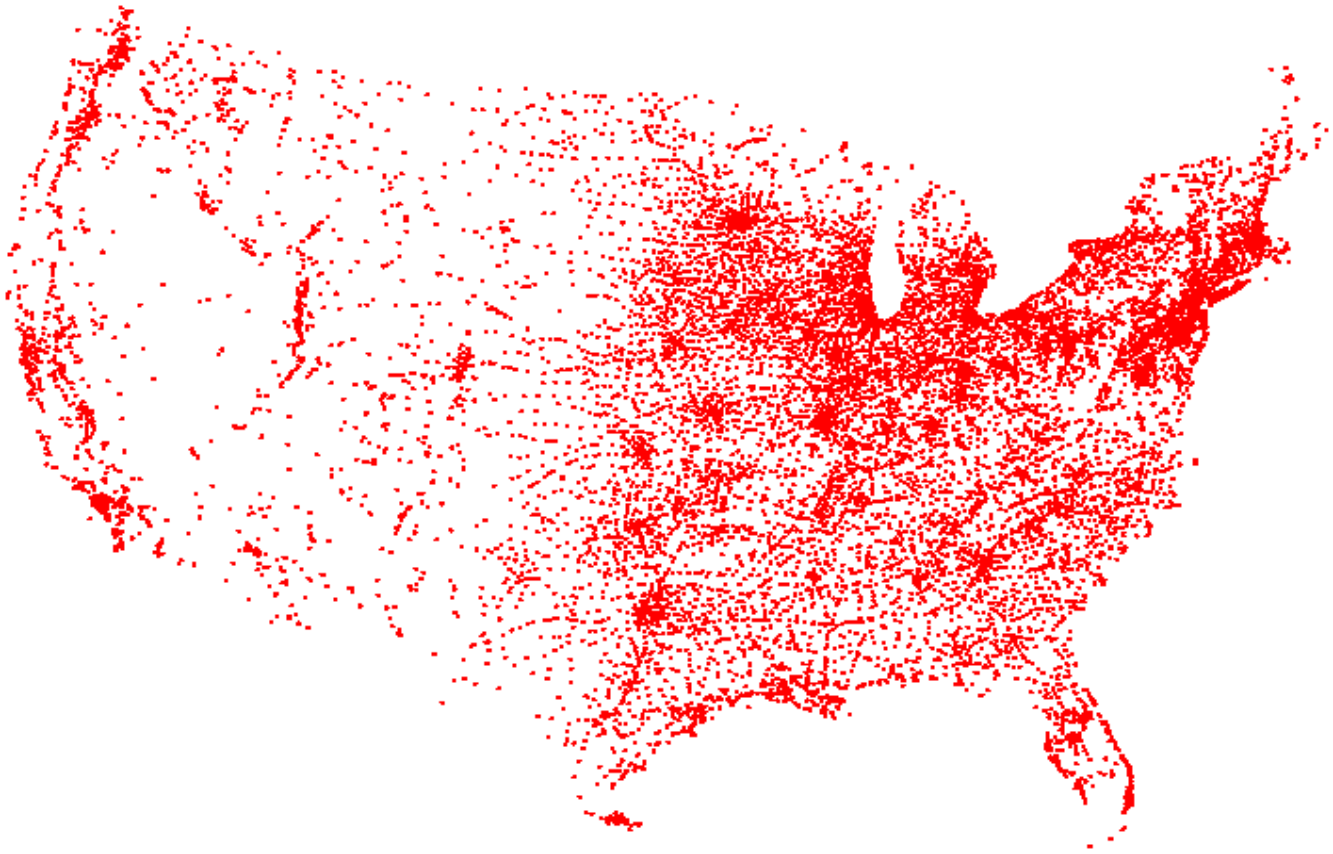
Garey, Johnson, Karp and other men (and women)
Tried to make it order N log N.
Am I a mad fool
If I spend my life in grad school,
Forever following the longest path?

Woh-oh-oh-oh, find the longest path!
Woh-oh-oh-oh, find the longest path!
Woh-oh-oh-oh, find the longest path.

† Recorded by Dan Barrett while a grad student at Johns Hopkins during a difficult algorithms final.

Traveling Salesperson Problem

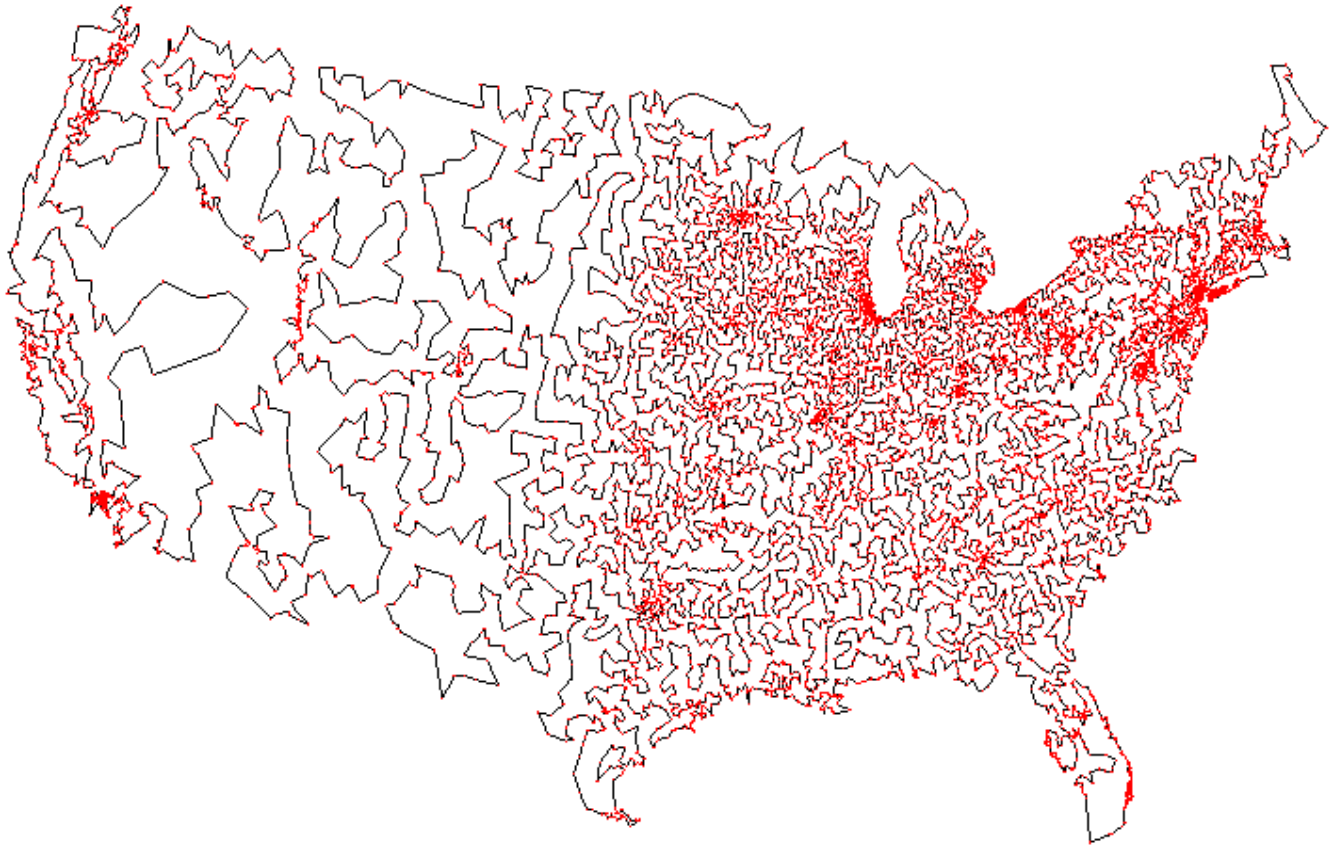
TSP. Given a set of n cities and a pairwise distance function d(u, v), is there a tour of length $\leq$ D?



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 $\leq$ D?



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

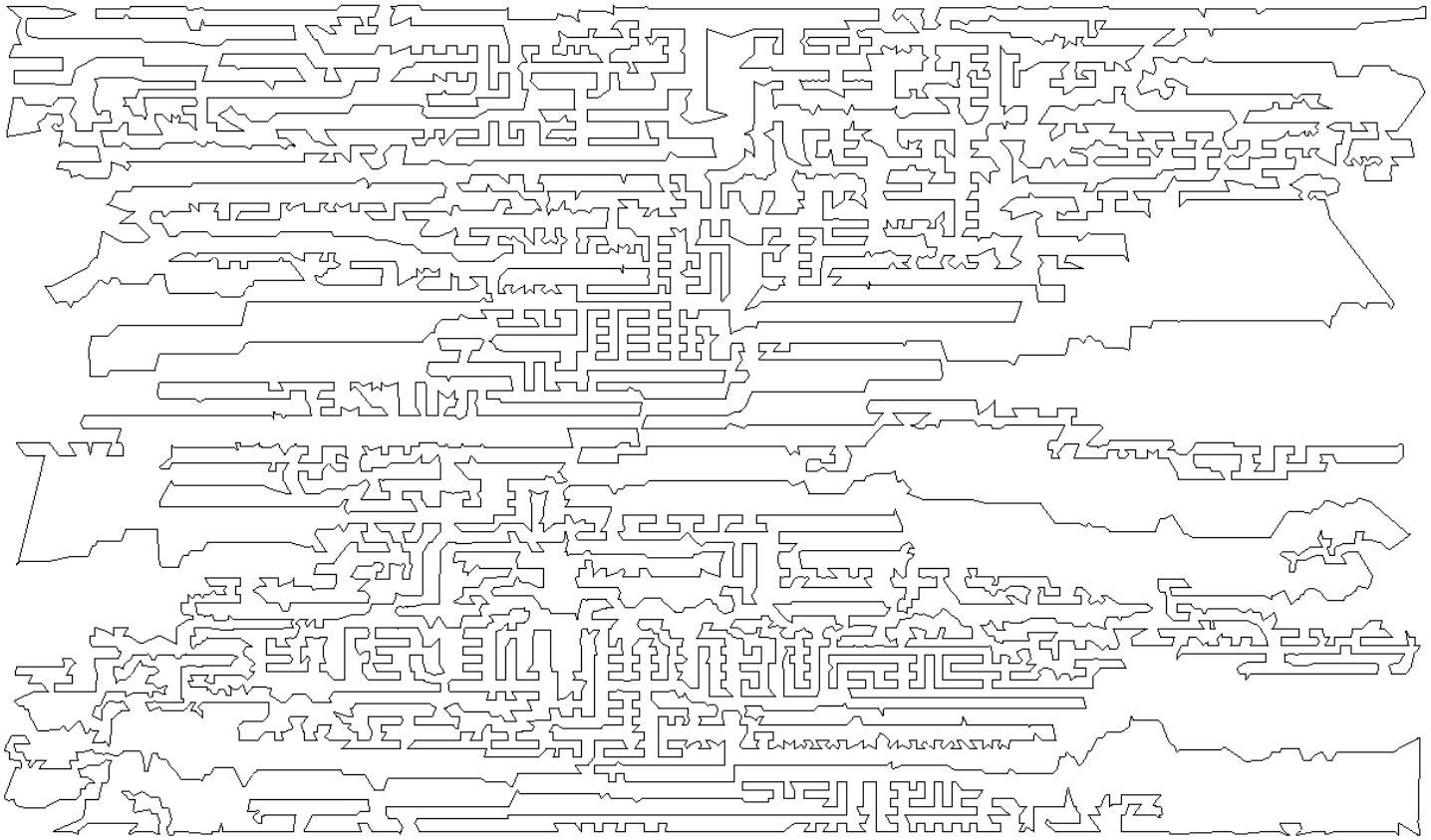
Traveling Salesperson Problem

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

11,849 holes to drill in a programmed logic array
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?



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) = 1 if (u,v) is in E, and d(u,v) = 2 otherwise.

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

Remark. TSP instance in reduction satisfies Δ-inequality.

8.6 Partitioning 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.

3-Dimensional Matching

3D-MATCHING. Given n instructors, n courses, and n times, and a list of the possible courses and times each instructor is willing to teach, is it possible to make an assignment so that all courses are taught at different times, by different instructors?

Instructor	Course	Time
Wayne	COS 423	MW 11-12:20
Wayne	COS 423	TTh 11-12:20
Wayne	COS 226	TTh 11-12:20
Wayne	COS 226	MW 11-12:20
Tardos	COS 523	TTh 3-4:20
Tardos	COS 423	TTh 11-12:20
Tardos	COS 423	TTh 3-4:20
Kleinberg	COS 226	TTh 3-4:20
Kleinberg	COS 226	MW 11-12:20
Kleinberg	COS 423	MW 11-12:20

3-Dimensional Matching

3D-MATCHING. Given 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?

Claim. 3-SAT $\leq_P$ 3D-matching.

Pf. Given an instance Φ of 3-SAT, we construct an instance of 3D-matching that has a perfect matching iff Φ is satisfiable.

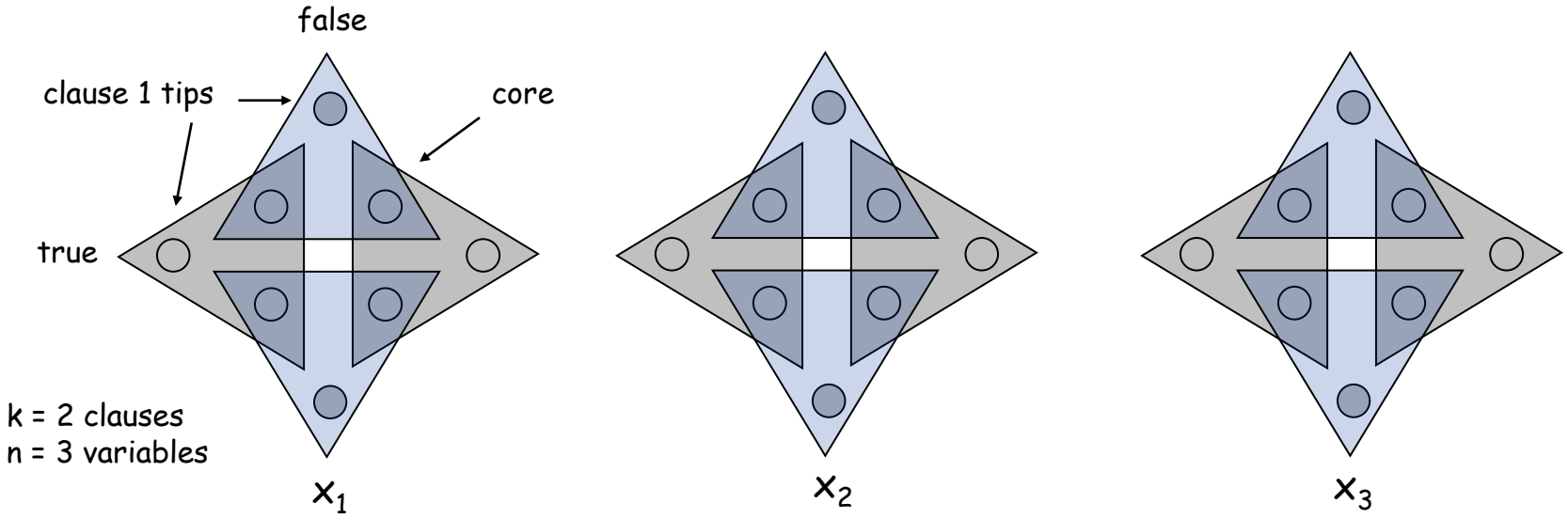
3-Dimensional Matching

Construction. (part 1)

number of clauses

- Create gadget for each variable x_i with 2k core and tip elements.
- No other triples will use core elements.
- In gadget i, 3D-matching must use either both grey triples or both blue ones.

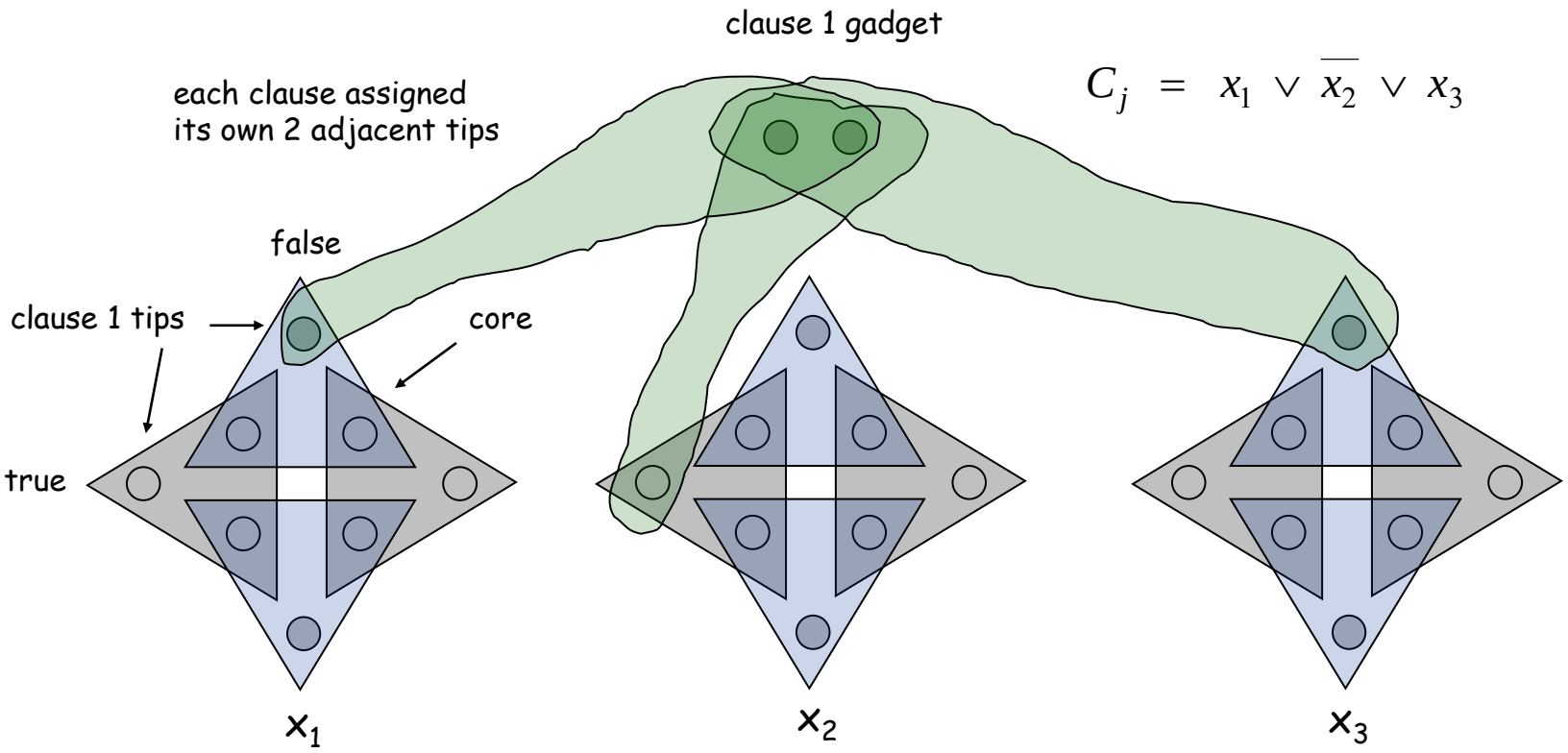
set x_i = true set x_i = false



3-Dimensional Matching

Construction. (part 2)

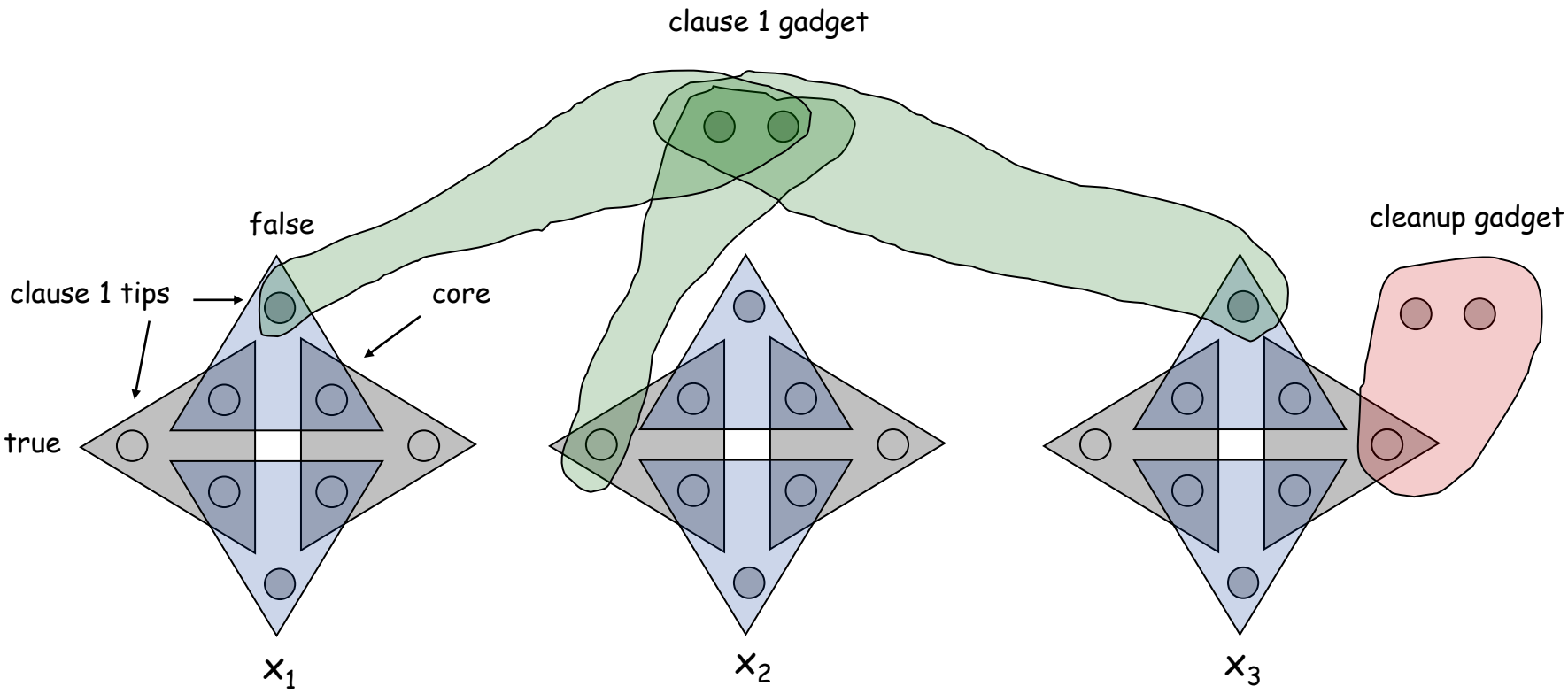
- For each clause C_j create two elements and three triples.
- Exactly one of these triples will be used in any 3D-matching.
- Ensures any 3D-matching uses either (i) grey core of x_1 or (ii) blue core of x_2 or (iii) grey core of x_3.



3-Dimensional Matching

Construction. (part 3)

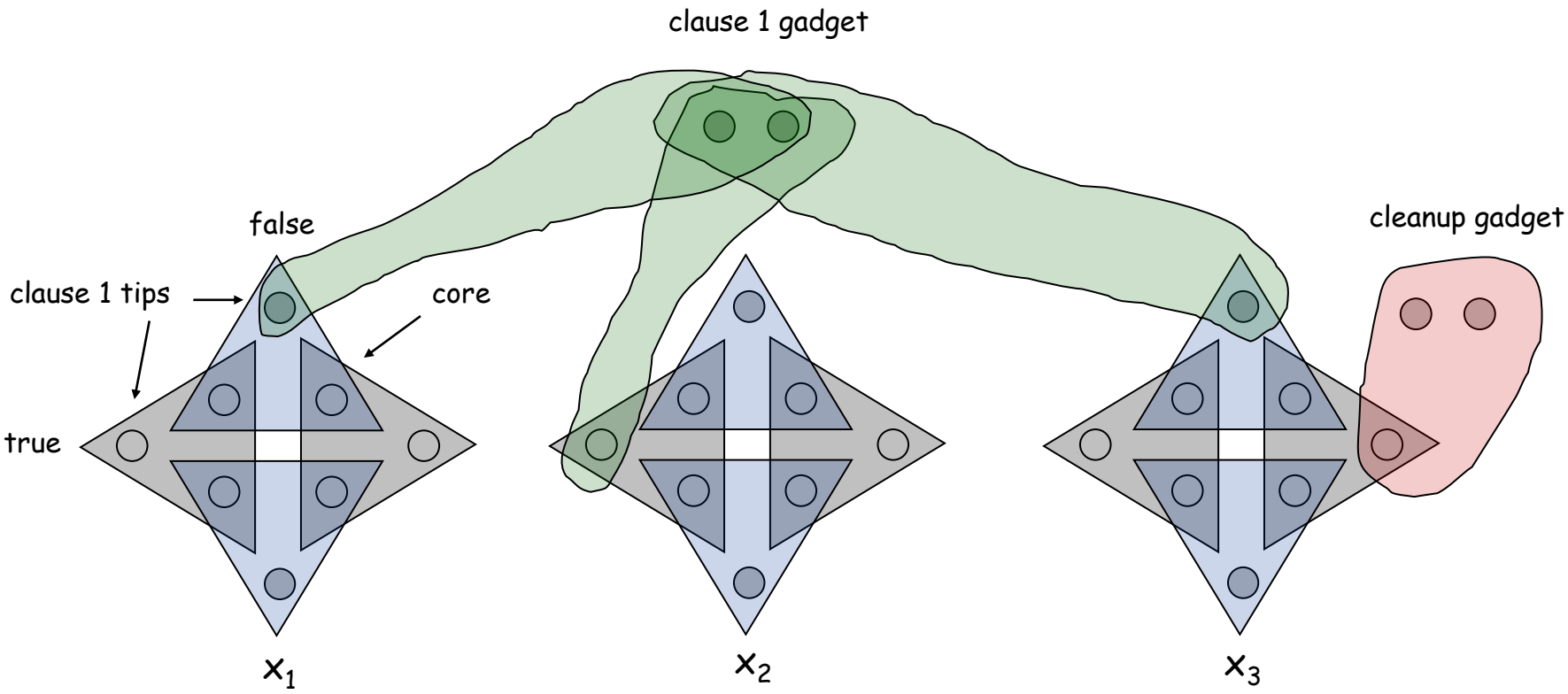
- For each tip, add a cleanup gadget.



3-Dimensional Matching

Claim. Instance has a 3D-matching iff Φ is satisfiable.

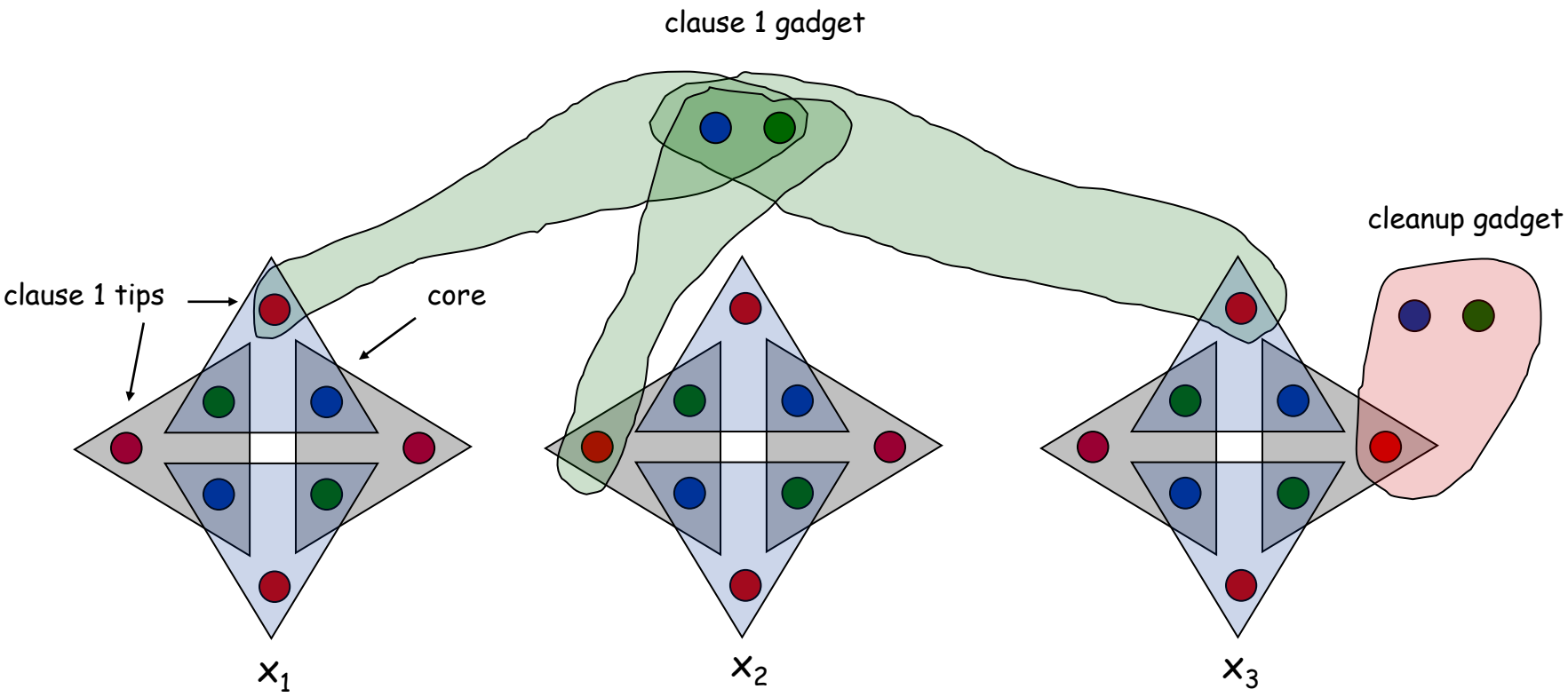
Detail. What are X, Y, and Z? Does each triple contain one element from each of X, Y, Z?



3-Dimensional Matching

Claim. Instance has a 3D-matching iff Φ is satisfiable.

Detail. What are X, Y, and Z? Does each triple contain one element from each of X, Y, Z?



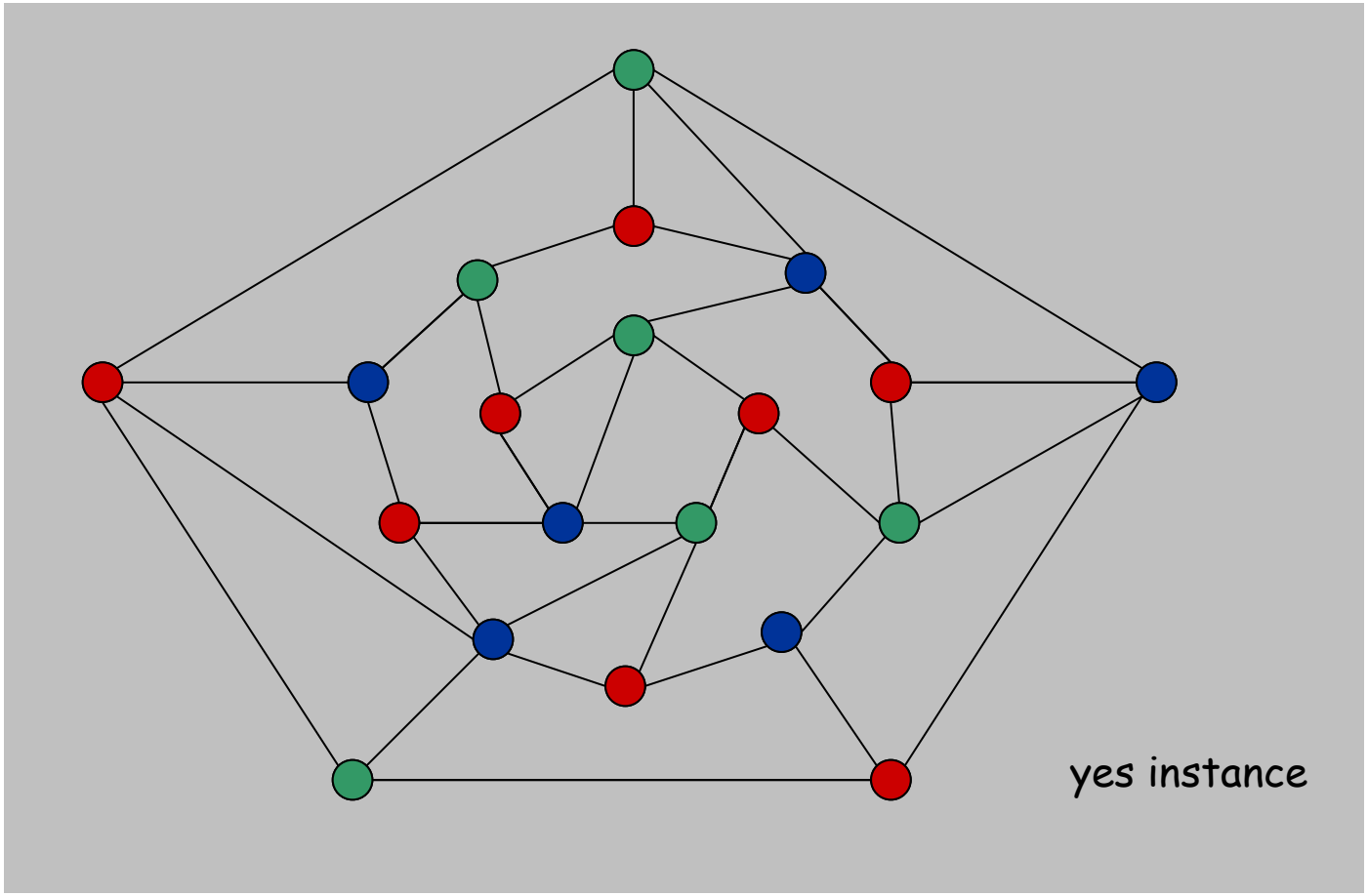
8.7 Graph Coloring

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.

3-Colorability

3-COLOR: Given an undirected graph G does there exists a way to color the nodes red, green, and blue so that no adjacent nodes have the same color?



Register Allocation

Register allocation. Assign program variables to machine register 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 names, 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 $k \geq 3$.

3-Colorability

Claim. 3-SAT $\leq_P$ 3-COLOR.

Pf. Given 3-SAT instance Φ, we construct an instance of 3-COLOR that is 3-colorable iff Φ is satisfiable.

Construction.

i. For each literal, create a node.

ii. Create 3 new nodes T, F, B; connect them in a triangle, and connect each literal to B.

iii. Connect each literal to its negation.

iv. For each clause, add gadget of 6 nodes and 13 edges.

↑

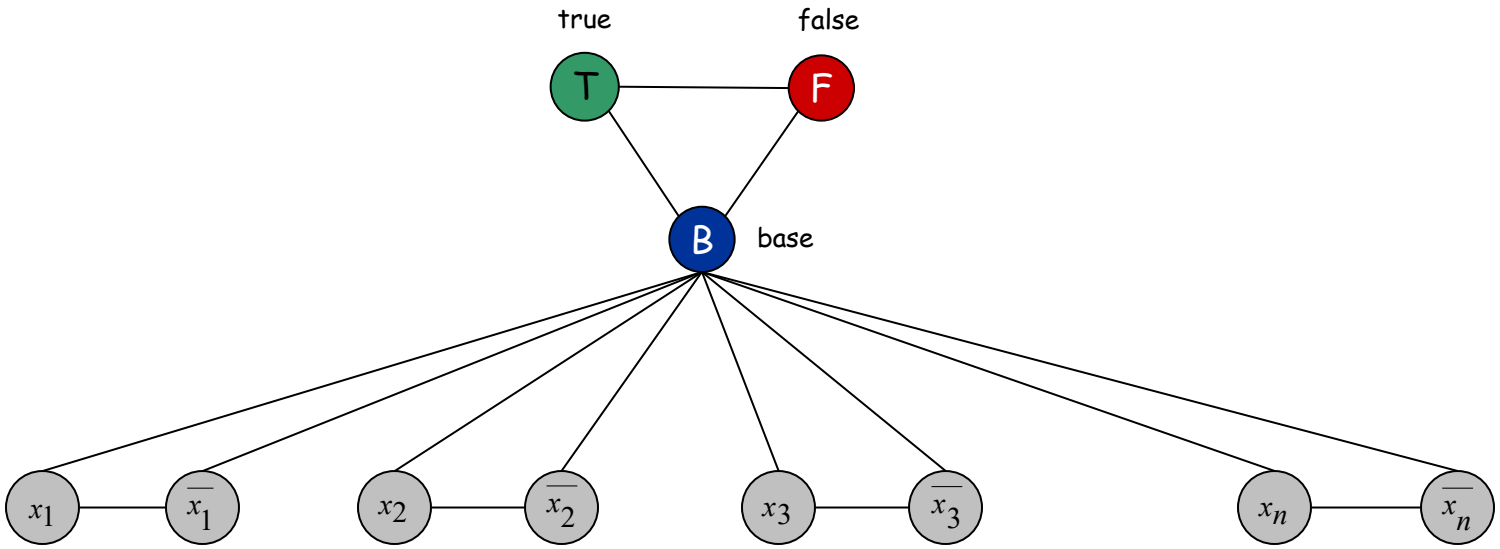
to be described next

3-Colorability

Claim. Graph is 3-colorable iff Φ is satisfiable.

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

- Consider assignment that sets all T literals to true.
- (ii) ensures each literal is T or F.
- (iii) ensures a literal and its negation are opposites.

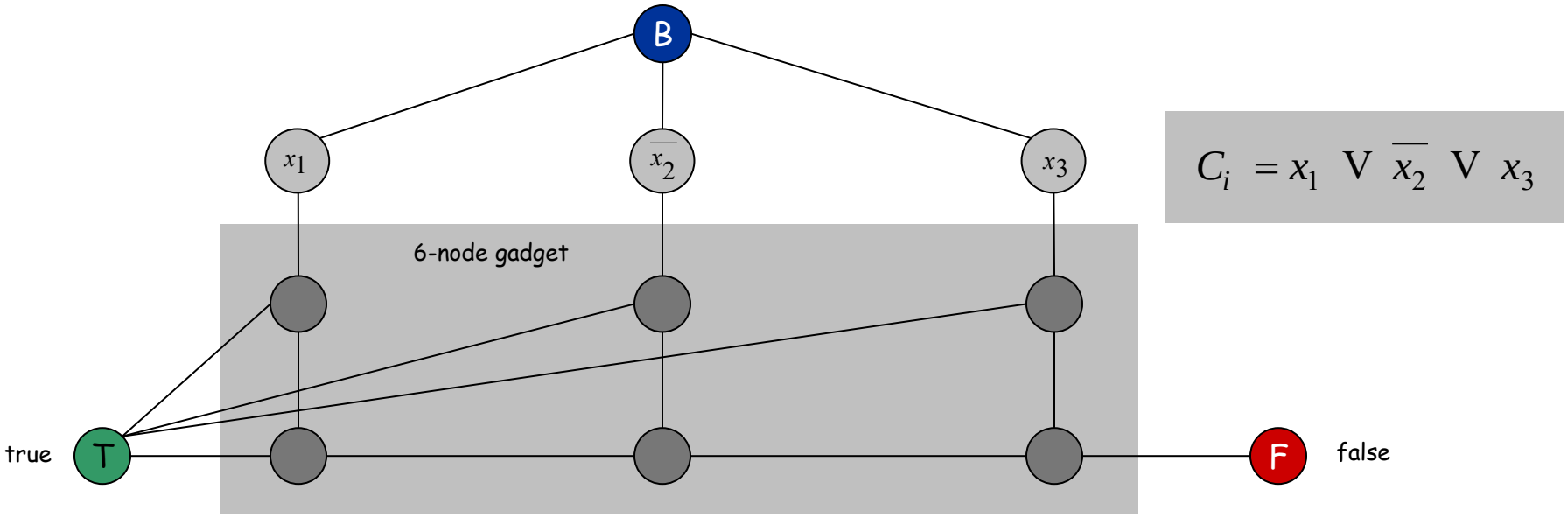


3-Colorability

Claim. Graph is 3-colorable iff Φ is satisfiable.

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

- Consider assignment that sets all T literals to true.
- (ii) ensures each literal is T or F.
- (iii) ensures a literal and its negation are opposites.
- (iv) ensures at least one literal in each clause is T.

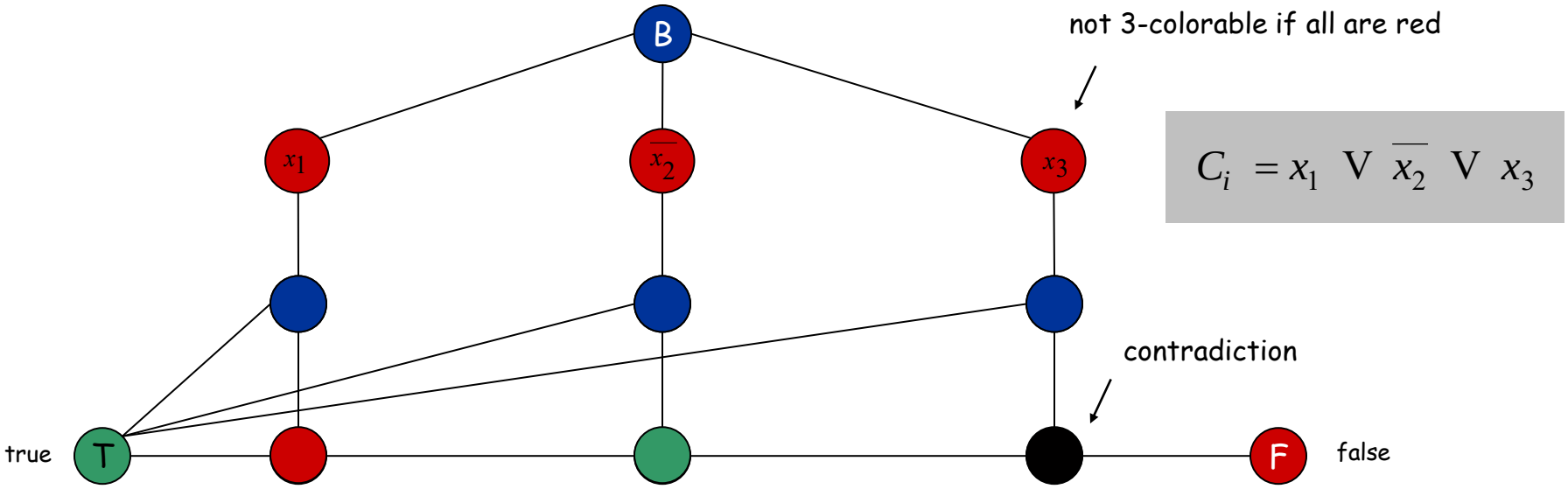


3-Colorability

Claim. Graph is 3-colorable iff Φ is satisfiable.

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

- Consider assignment that sets all T literals to true.
- (ii) ensures each literal is T or F.
- (iii) ensures a literal and its negation are opposites.
- (iv) ensures at least one literal in each clause is T.

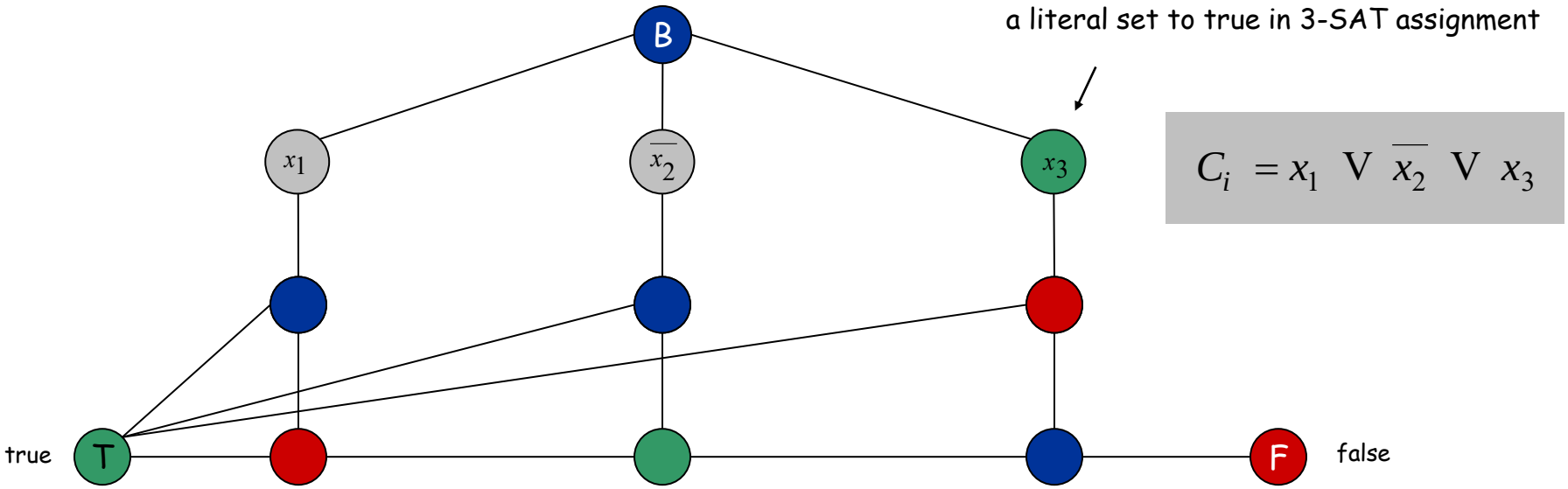


3-Colorability

Claim. Graph is 3-colorable iff Φ is satisfiable.

Pf. $\Leftarrow$ Suppose 3-SAT formula Φ is satisfiable.

- Color all true literals T.
- Color node below green node F, and node below that B.
- Color remaining middle row nodes B.
- Color remaining bottom nodes T or F as forced. ▪



8.8 Numerical 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: 3-COLOR, 3D-MATCHING.
- Numerical problems: SUBSET-SUM, KNAPSACK.

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. Polynomial reduction must be polynomial in binary encoding.

Claim. 3-SAT $\leq_P$ SUBSET-SUM.
Pf. Given an instance Φ of 3-SAT, we construct an instance of SUBSET-SUM that has solution iff Φ is satisfiable.

Subset Sum

Construction. Given 3-SAT instance Φ with n variables and k clauses, form 2n + 2k decimal integers, each of n+k digits, as illustrated below.

Claim. Φ is satisfiable iff there exists a subset that sums to W.

Pf. No carries possible.

$$C_1 = \neg x_1 \vee x_2 \vee x_3$$
$$C_2 = x_1 \vee \neg x_2 \vee x_3$$
$$C_3 = \neg x_1 \vee \neg x_2 \vee \neg x_3$$

3-SAT instance

	x_1	x_2	x_3	C_1	C_2	C_3	
x_1	1	0	0	0	1	0	100,010
$\neg x_1$	1	0	0	1	0	1	100,101
x_2	0	1	0	1	0	0	10,100
$\neg x_2$	0	1	0	0	1	1	10,011
x_3	0	0	1	1	1	0	1,110
$\neg x_3$	0	0	1	0	0	1	1,001
dummies to get clause columns to sum to 4	0	0	0	1	0	0	100
	0	0	0	2	0	0	200
	0	0	0	0	1	0	10
	0	0	0	0	2	0	20
	0	0	0	0	0	1	1
	0	0	0	0	0	2	2
W	1	1	1	4	4	4	111,444

Scheduling With Release Times

SCHEDULE-RELEASE-TIMES. Given a set of n jobs with processing time t_i, release time r_i, and deadline d_i, is it possible to schedule all jobs on a single machine such that job i is processed with a contiguous slot of t_i time units in the interval $[r_i, d_i]$?

Claim. SUBSET-SUM $\leq_P$ SCHEDULE-RELEASE-TIMES.

Pf. Given an instance of SUBSET-SUM $w_1, \ldots, w_n$, and target W,

- Create n jobs with processing time $t_i = w_i$, release time $r_i = 0$, and no deadline ($d_i = 1 + \Sigma_j w_j$).
- Create job 0 with $t_0 = 1$, release time $r_0 = W$, and deadline $d_0 = W+1$.