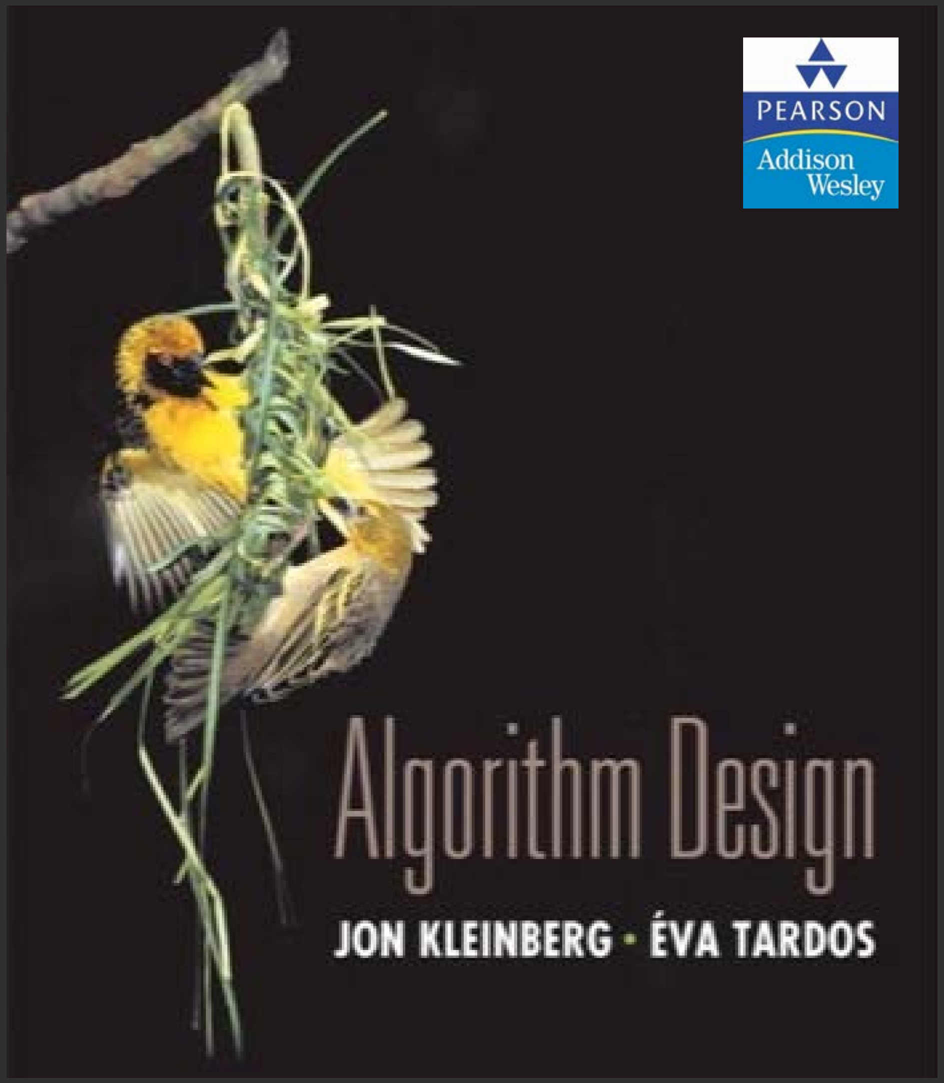


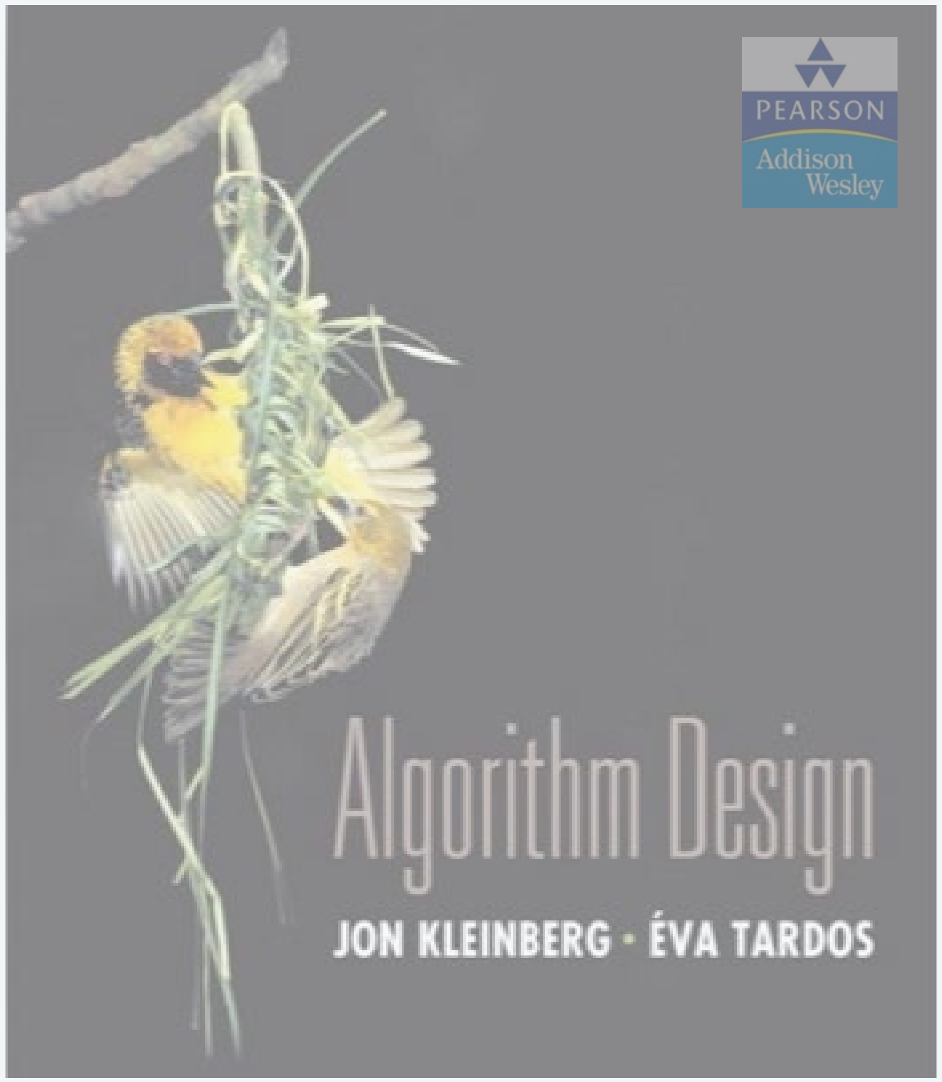
8. Intractability I

- *poly-time reductions*
- *packing and covering problems*
- *constraint satisfaction problems*
- *sequencing problems*
- *partitioning problems*
- *graph coloring*
- *numerical problems*

Lecture slides by Kevin Wayne

http://www.cs.princeton.edu/~wayne/kleinberg-tardos

Last updated on 15/9/12 下午10:44



8. Intractability I

- *poly-time reductions*
- *packing and covering problems*
- *constraint satisfaction problems*
- *sequencing problems*
- *partitioning problems*
- *graph coloring*
- *numerical problems*

Algorithm design patterns and antipatterns

Algorithm design patterns.

- Greedy.
- Divide and conquer.
- Dynamic programming.
- Duality.
- Reductions.
- Local search.
- Randomization.

Algorithm design antipatterns.

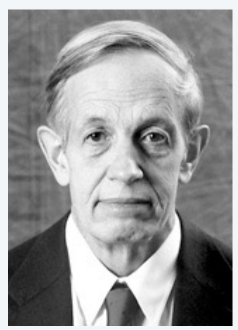
- NP-completeness. $O(n^k)$ algorithm unlikely.
- PSPACE-completeness. $O(n^k)$ certification algorithm unlikely.
- Undecidability. No algorithm possible.

Classify problems according to computational requirements

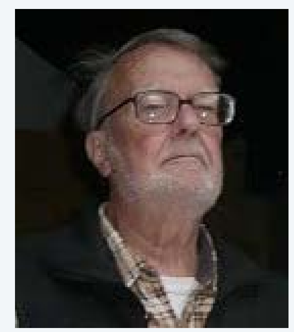
Q. Which problems will we be able to solve in practice?

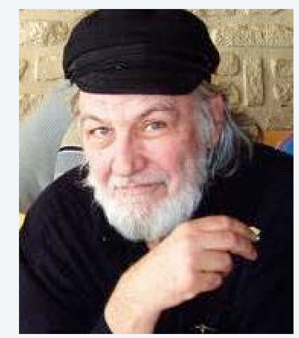
A working definition. Those with polynomial-time algorithms.


von Neumann
(1953)


Nash
(1955)


Gödel
(1956)


Cobham
(1964)


Edmonds
(1965)


Rabin
(1966)

Theory. Definition is broad and robust.

constants a and b tend to be small, e.g., $3\ N^{2}$

Practice. Poly-time algorithms scale to huge problems.

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.

yes	probably no
shortest path	longest path
min cut	max cut
2-satisfiability	3-satisfiability
planar 4-colorability	planar 3-colorability
bipartite vertex cover	vertex cover
matching	3d-matching
primality testing	factoring
linear programming	integer linear programming

Classify problems

Desiderata. Classify problems according to those that can be solved in polynomial time and those that cannot.

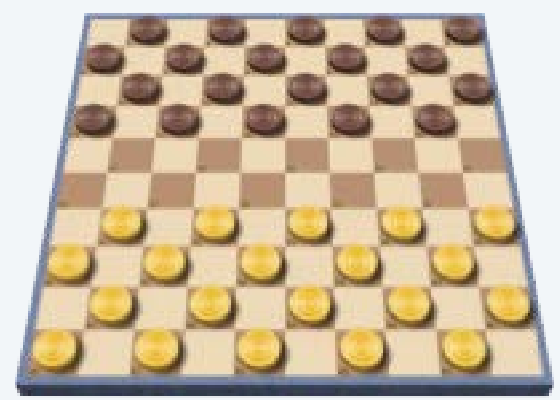
Provably requires exponential time.

input size = c + lg k

- Given a constant-size program, does it halt in at most k steps?
- Given a board position in an n-by-n generalization of checkers, can black guarantee a win?

using forced capture rule

Alan designed the perfect computer



Frustrating news. Huge number of fundamental problems have defied classification for decades.

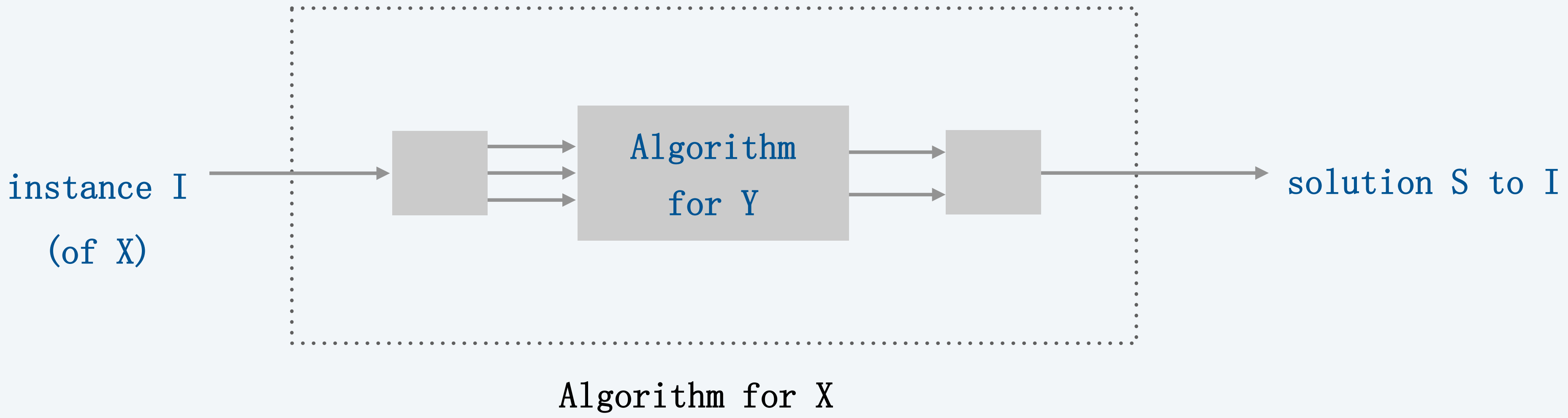
Polynomial-time reductions

Desiderata'. Suppose we could solve X 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.

computational model supplemented by special piece of hardware that solves instances of Y in a single step



Desiderata'. Suppose we could solve X 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 ® instances of Y must be of polynomial size.

Caveat. Don't mistake $X \leq_P Y$ with $Y \leq_P X$.

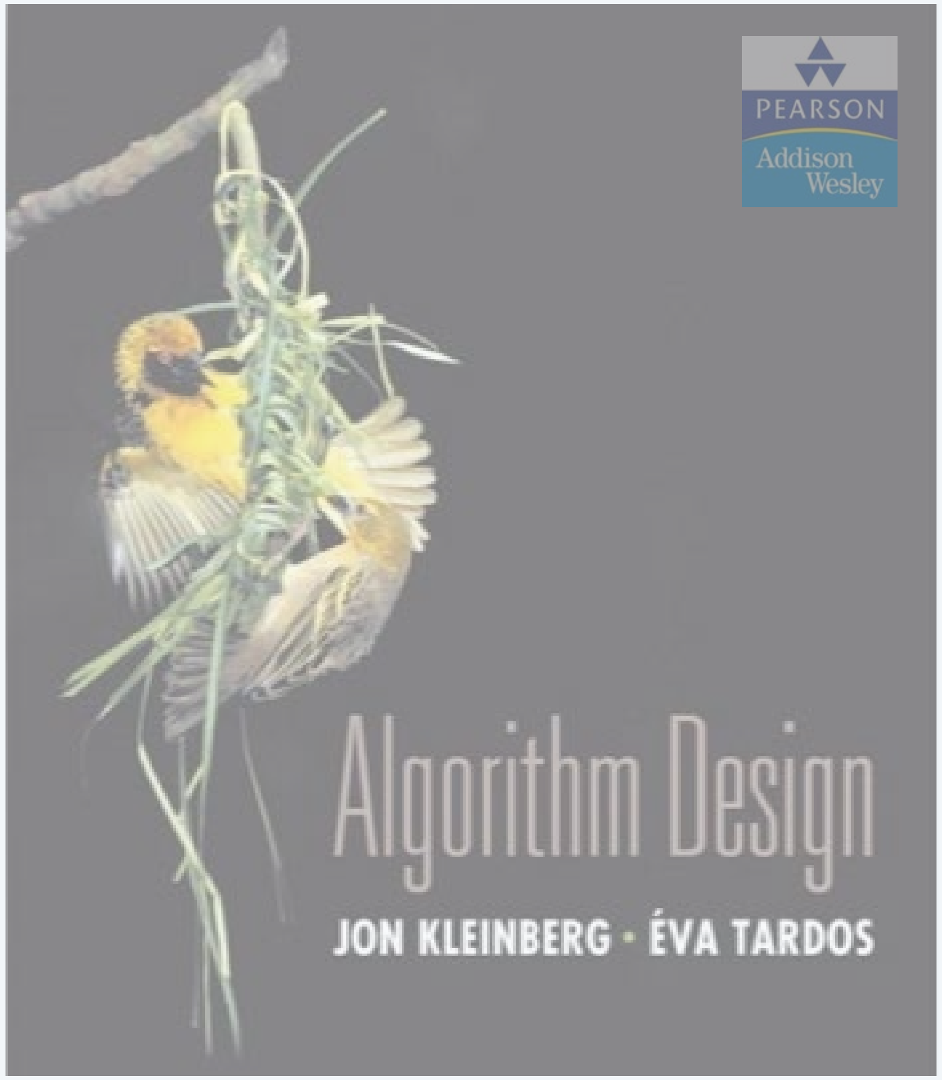
Polynomial-time reductions

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.

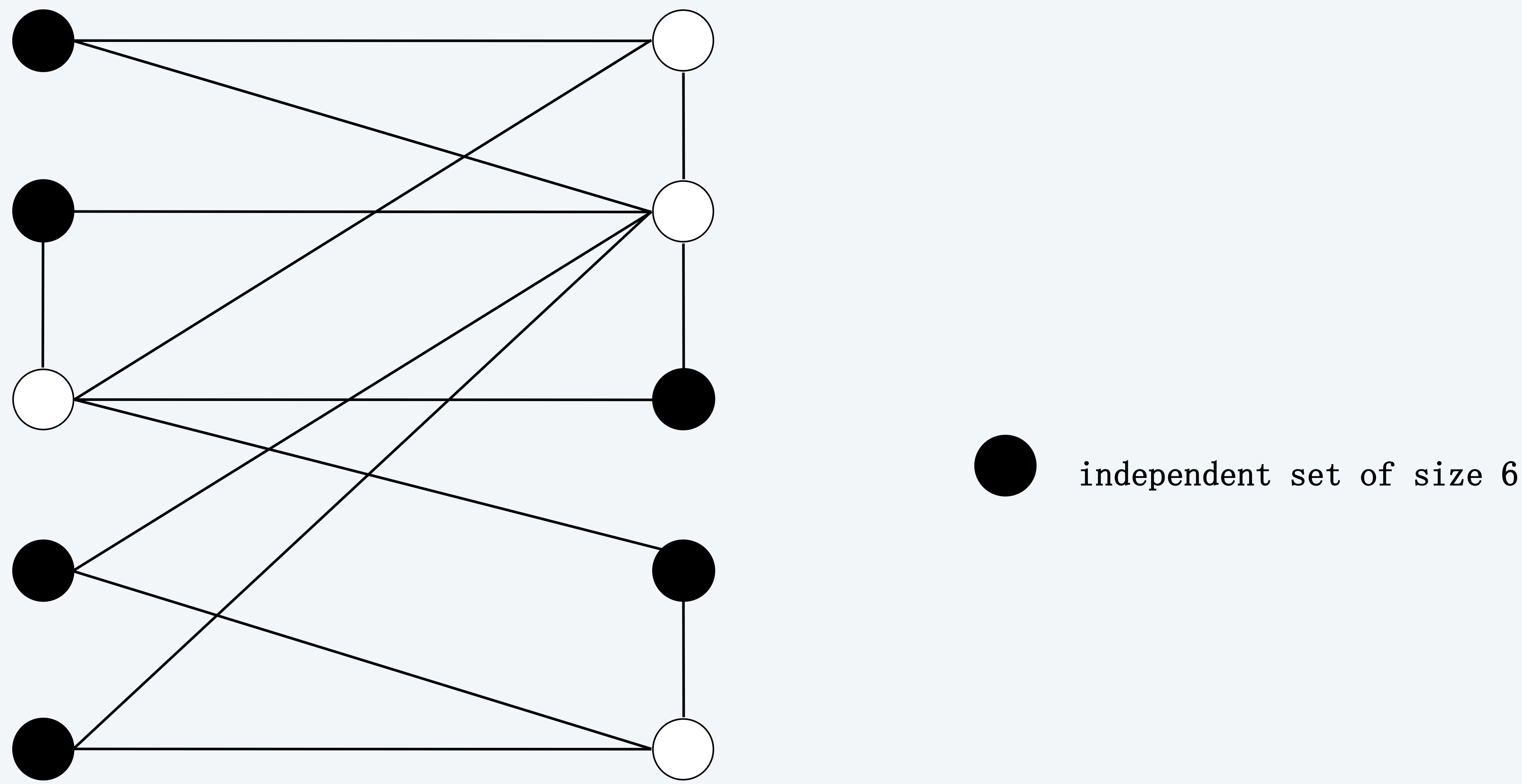


8. INTRACTABILITY I

- *poly-time reductions*
- *packing and covering problems*
- *constraint satisfaction problems*
- *sequencing problems*
- *partitioning problems*
- *graph coloring*
- *numerical problems*

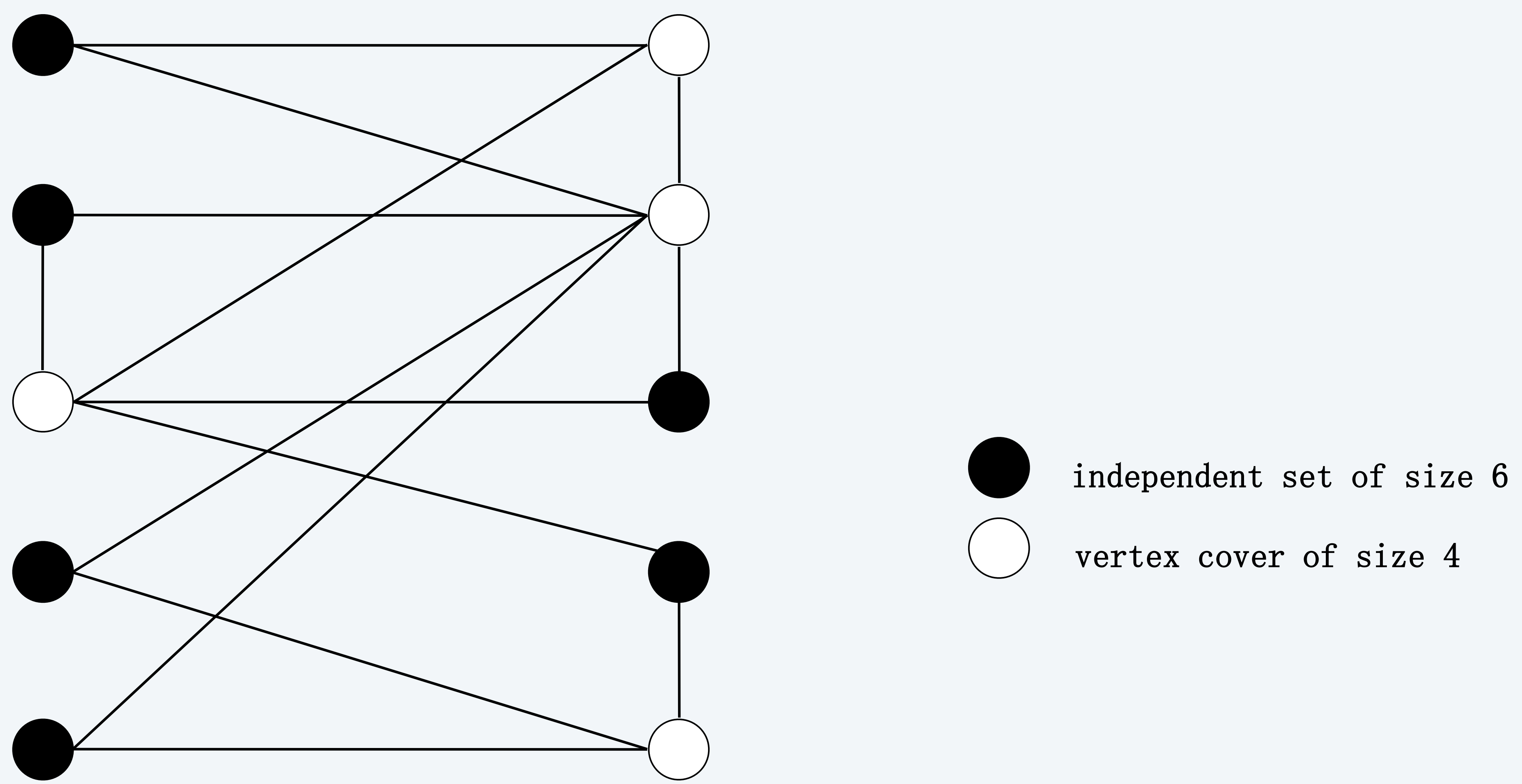
INDEPENDENT-SET. Given a graph $G = (V, E)$ and an integer k, is there a subset of vertices $S \subseteq V$ such that $|S| \geq k$, and for each edge at most one of its endpoints is in S?

Ex. Is there an independent set of size ≥ 6?
Ex. Is there an independent set of size ≥ 7?



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?

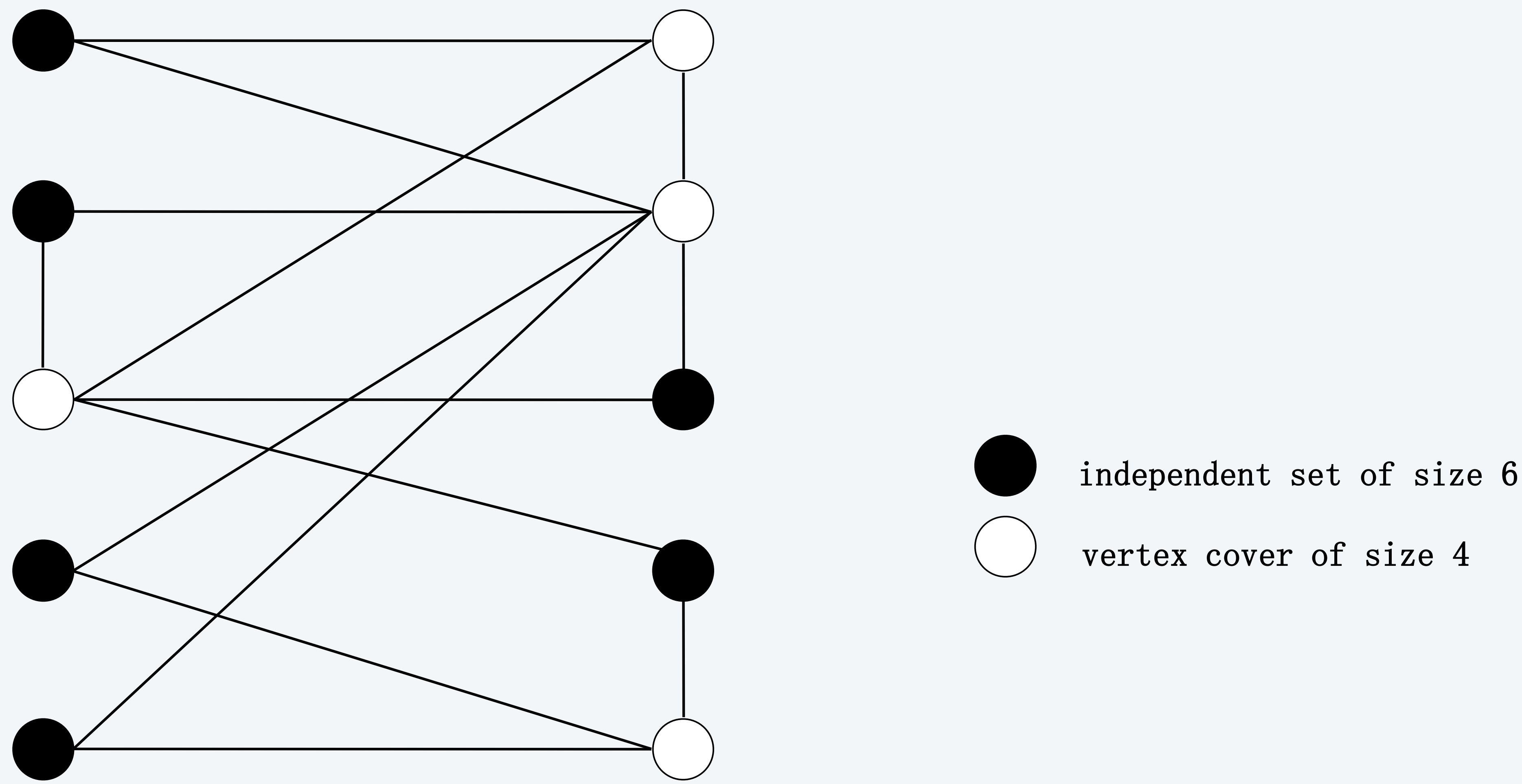
Ex. Is there a vertex cover of size ≤ 4?
Ex. Is there a vertex cover of size ≤ 3?



Vertex cover and independent set reduce to one another

Theorem. VERTEX-COVER $\equiv_P$ INDEPENDENT-SET.

Pf. We show S is an independent set of size k iff $V - S$ is a vertex cover of size $n - k$.



Vertex cover and independent set reduce to one another

Theorem. VERTEX-COVER $\equiv_P$ INDEPENDENT-SET.

Pf. We show S is an independent set of size k iff $V - S$ is a vertex cover of size $n - k$.

®

- 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. VERTEX-COVER $\equiv_P$ INDEPENDENT-SET.

Pf. We show S is an independent set of size k iff $V - S$ is a vertex cover of size $n - k$.

$\angle$

- 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 ® S independent set. ▪

SET-COVER. Given a set U of elements, a collection $S_1, S_2, \ldots, S_m$ of subsets of U, and an integer k, does there exist a collection of $\leq k$ of these sets 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.

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

$S_1 = \{ 3, 7 \}$	$S_4 = \{ 2, 4 \}$
$S_2 = \{ 3, 4, 5, 6 \}$	$S_5 = \{ 5 \}$
$S_3 = \{ 1 \}$	$S_6 = \{ 1, 2, 6, 7 \}$

$k = 2$

a set cover instance

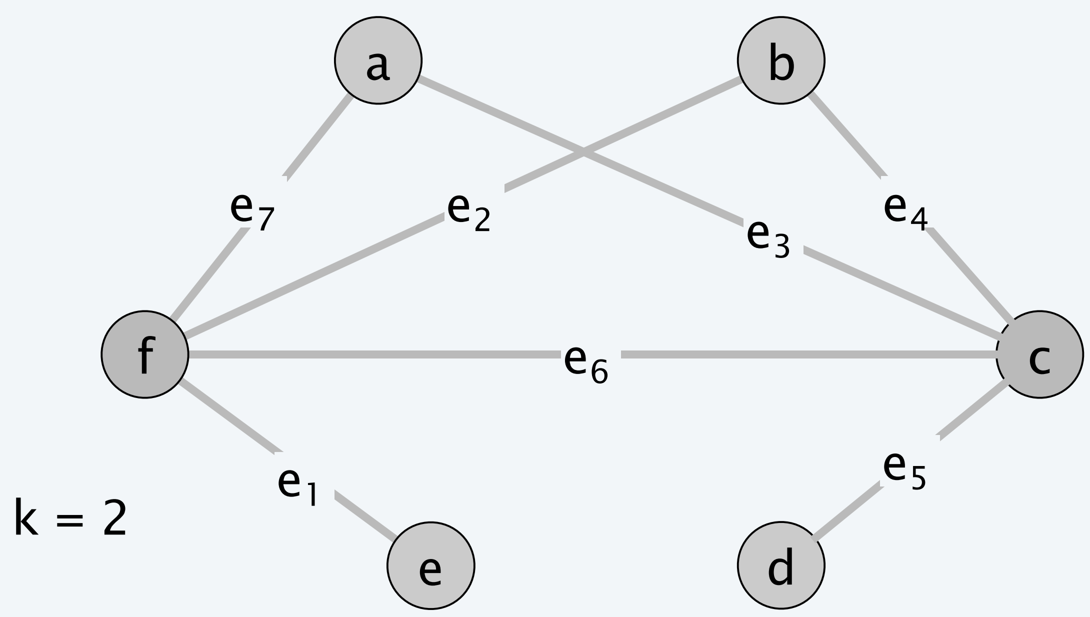
Vertex cover reduces to set cover

Theorem. VERTEX-COVER $\leq_P$ SET-COVER.

Pf. Given a VERTEX-COVER instance $G = (V, E)$, we construct a 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 set for each node $v \in V$: $S_v = \{e \in E : e \text{ incident to } v\}$.



vertex cover instance
(k = 2)

$U = \{ 1, 2, 3, 4, 5, 6, 7 \}$
$S_a = \{ 3, 7 \}$ $\quad S_b = \{ 2, 4 \}$
$S_c = \{ 3, 4, 5, 6 \}$ $\quad S_d = \{ 5 \}$
$S_e = \{ 1 \}$ $\quad S_f = \{ 1, 2, 6, 7 \}$

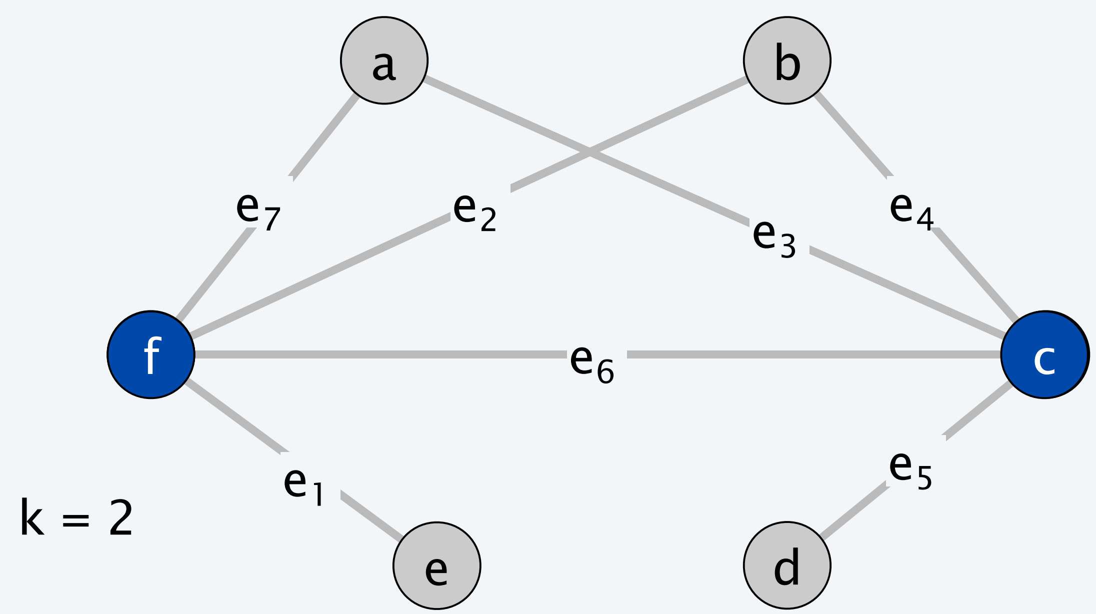
set cover instance
(k = 2)

Vertex cover reduces to set cover

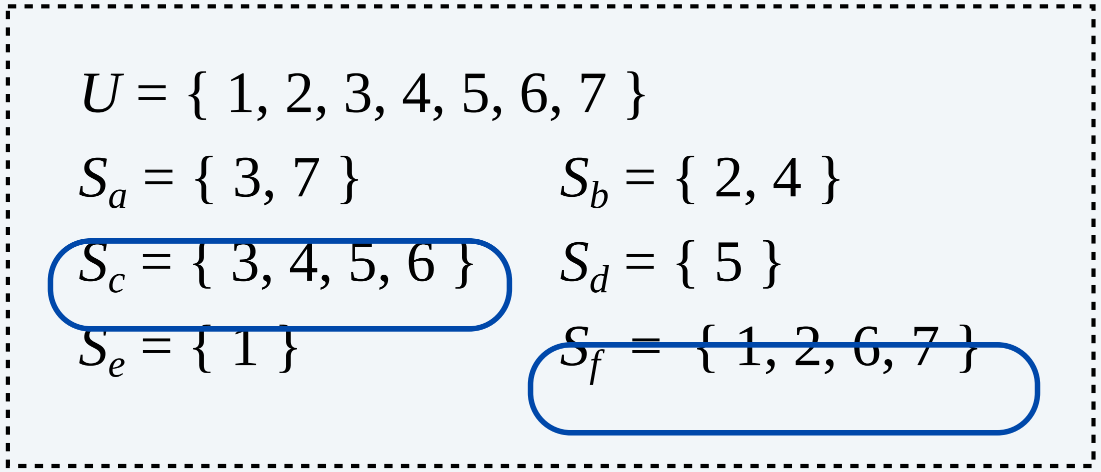
Lemma. $G = (V, E)$ contains a vertex cover of size k iff (U, S) contains a set cover of size k.

Pf. ® 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)



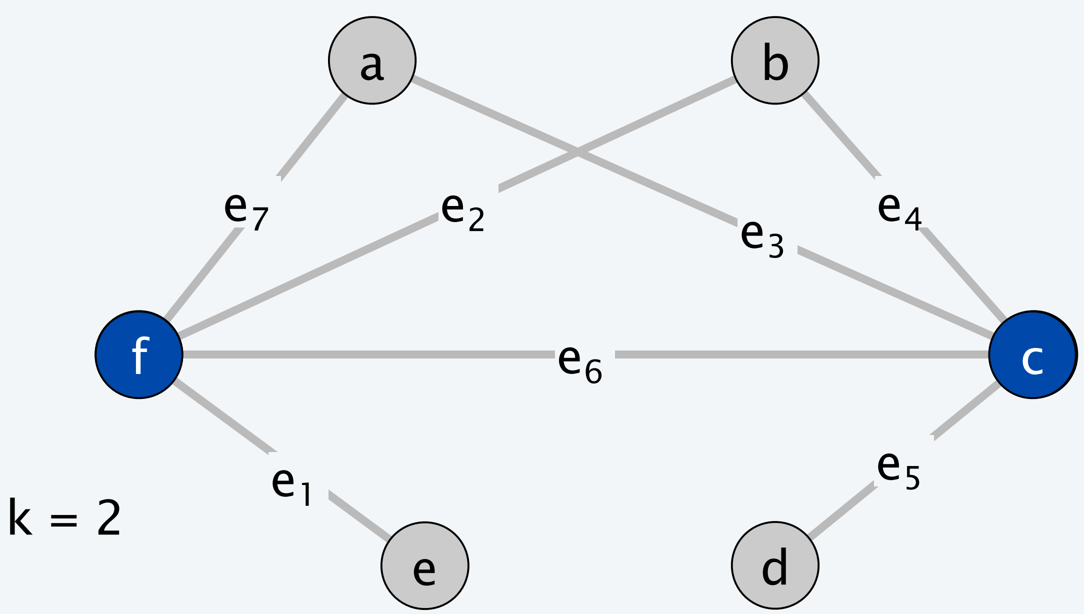
set cover instance
(k = 2)

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. ∠ 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. ▪



vertex cover instance
(k = 2)

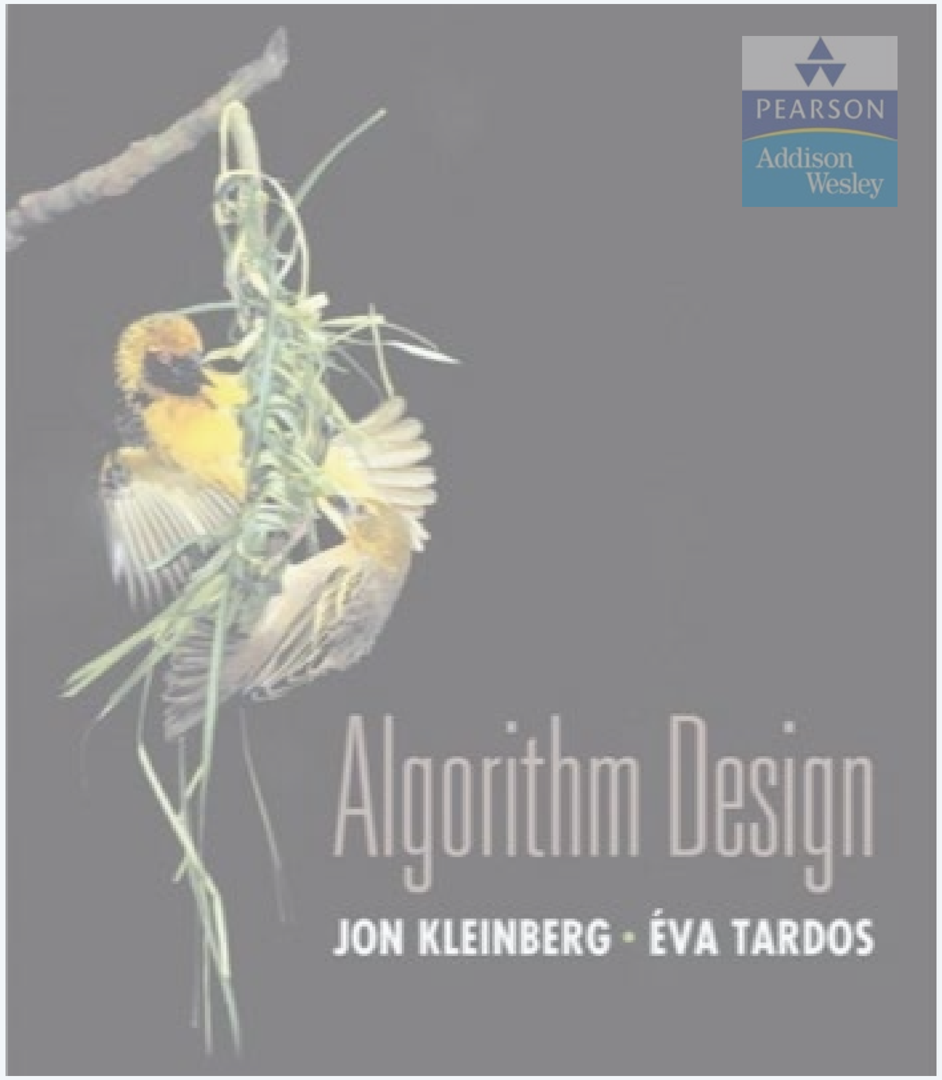
$U = \{ 1, 2, 3, 4, 5, 6, 7 \}$

$S_a = \{ 3, 7 \}$ $\quad$ $S_b = \{ 2, 4 \}$

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

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

set cover instance
(k = 2)



8. INTRACTABILITY I

- *poly-time reductions*
- *packing and covering problems*
- *constraint satisfaction problems*
- *sequencing problems*
- *partitioning problems*
- *graph coloring*
- *numerical problems*

Literal. A boolean variable or its negation.

$$x_i \text{ or } \overline{x_i}$$

Clause. A disjunction of literals.

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

Conjunctive normal form. A propositional formula Φ that is the conjunction of clauses.

$$\Phi = C_1 \wedge C_2 \wedge C_3 \wedge C_4$$

SAT. Given CNF formula Φ, 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).

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

yes instance: x_1 = true, x_2 = true, x_3 = false, x_4 = false

Key application. Electronic design automation (EDA).

3-satisfiability reduces to independent set

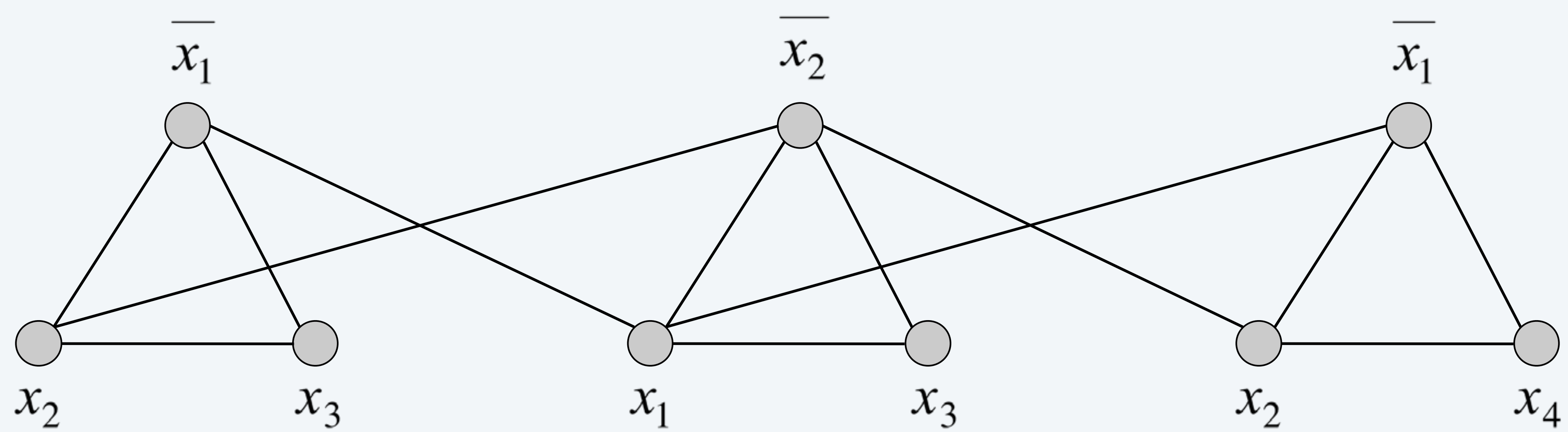
Theorem. 3-SAT $\leq_P$ INDEPENDENT-SET.

Pf. Given an instance Φ of 3-SAT, we construct an instance (G, k) of INDEPENDENT-SET that has an independent set of size k iff Φ 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.

G



k = 3

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

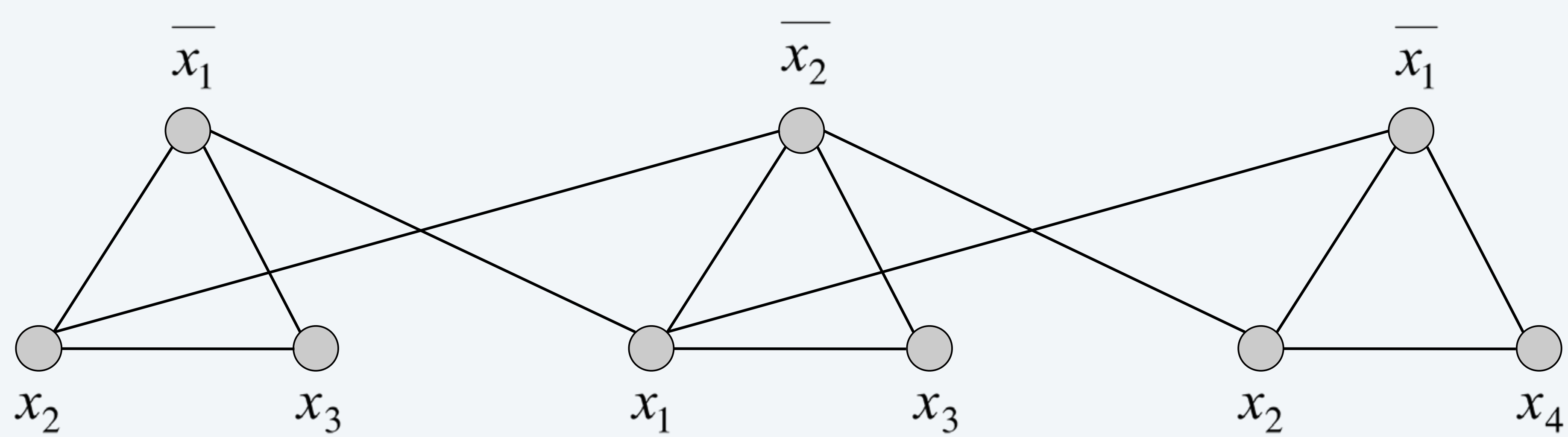
Lemma. G contains independent set of size $k = |\Phi|$ iff Φ is satisfiable.

Pf. ® 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 ∠ Given satisfying assignment, select one true literal from each triangle. This is an independent set of size k. ▪

G



k = 3

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

Basic reduction strategies.

- Simple equivalence: INDEPENDENT-SET $\equiv_P$ VERTEX-COVER.
- Special case to general case: VERTEX-COVER $\leq_P$ SET-COVER.
- Encoding with gadgets: 3-SAT $\leq_P$ INDEPENDENT-SET.

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

Pf idea. Compose the two algorithms.

Ex. 3-SAT δ_P INDEPENDENT-SET $\leq_P$ VERTEX-COVER $\leq_P$ SET-COVER.

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 $G - \{ v \}$.

delete v and all incident edges

Bottom line. VERTEX-COVER $\equiv_P$ FIND-VERTEX-COVER.

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

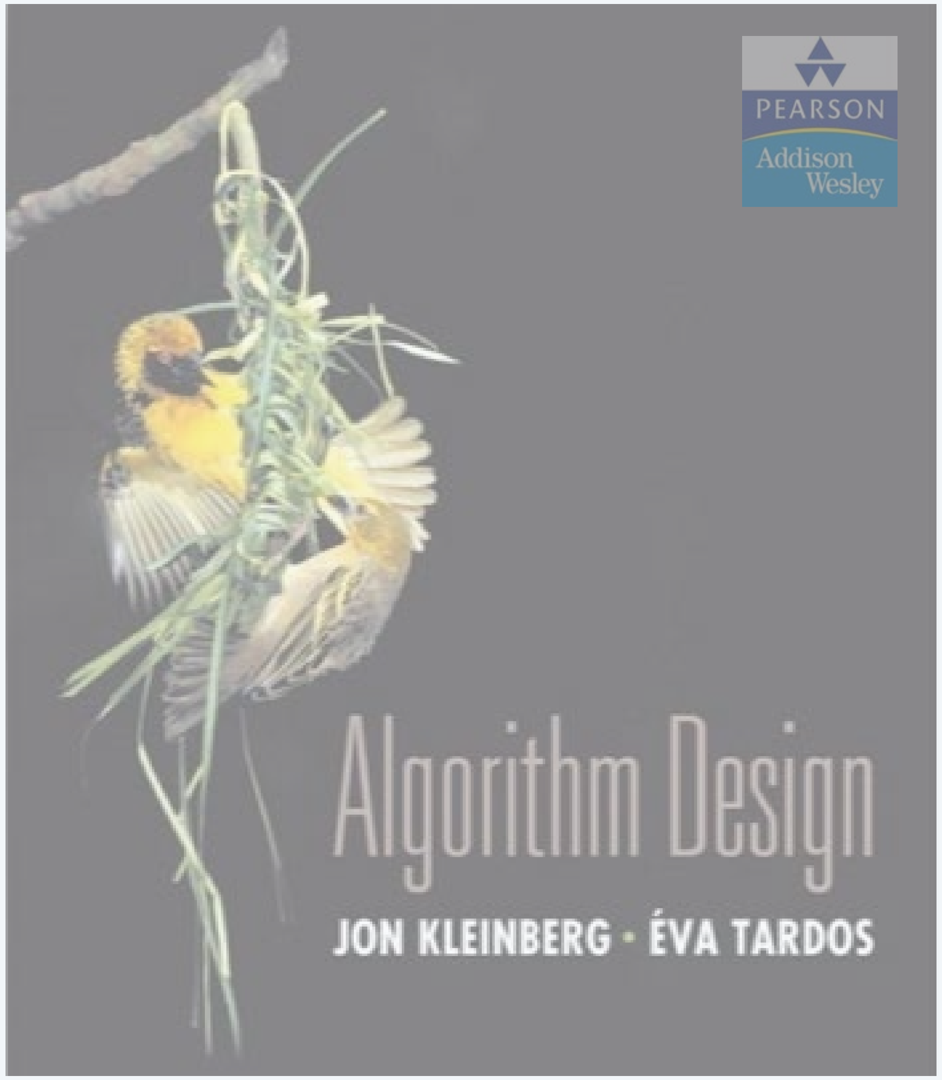
Search problem. Find a vertex cover of size $\leq k$.

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.

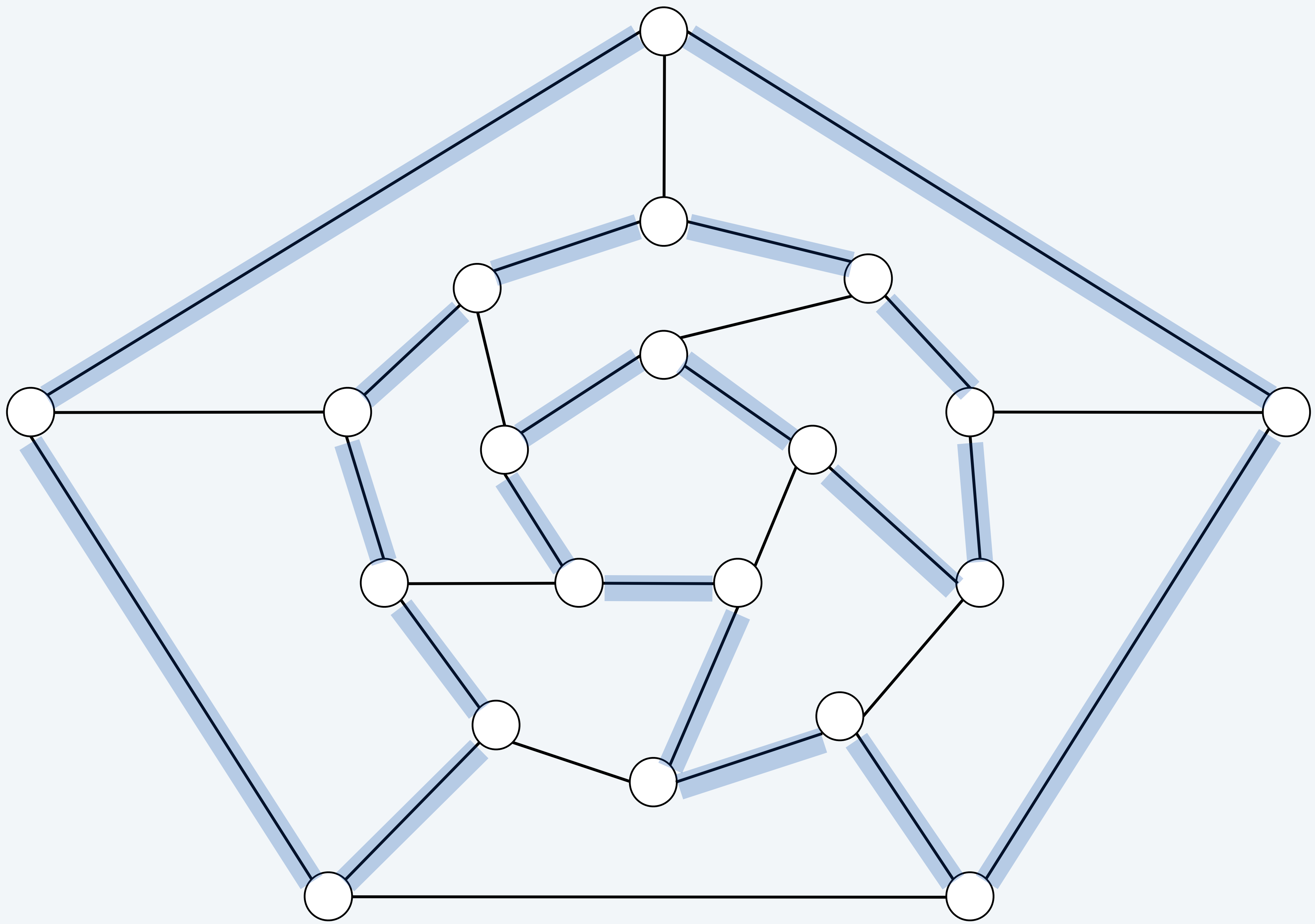
Bottom line. VERTEX-COVER $\equiv_P$ FIND-VERTEX-COVER $\equiv_P$ OPTIMAL-VERTEX-COVER.



8. INTRACTABILITY I

- *poly-time reductions*
- *packing and covering problems*
- *constraint satisfaction problems*
- *sequencing problems*
- *partitioning problems*
- *graph coloring*
- *numerical problems*

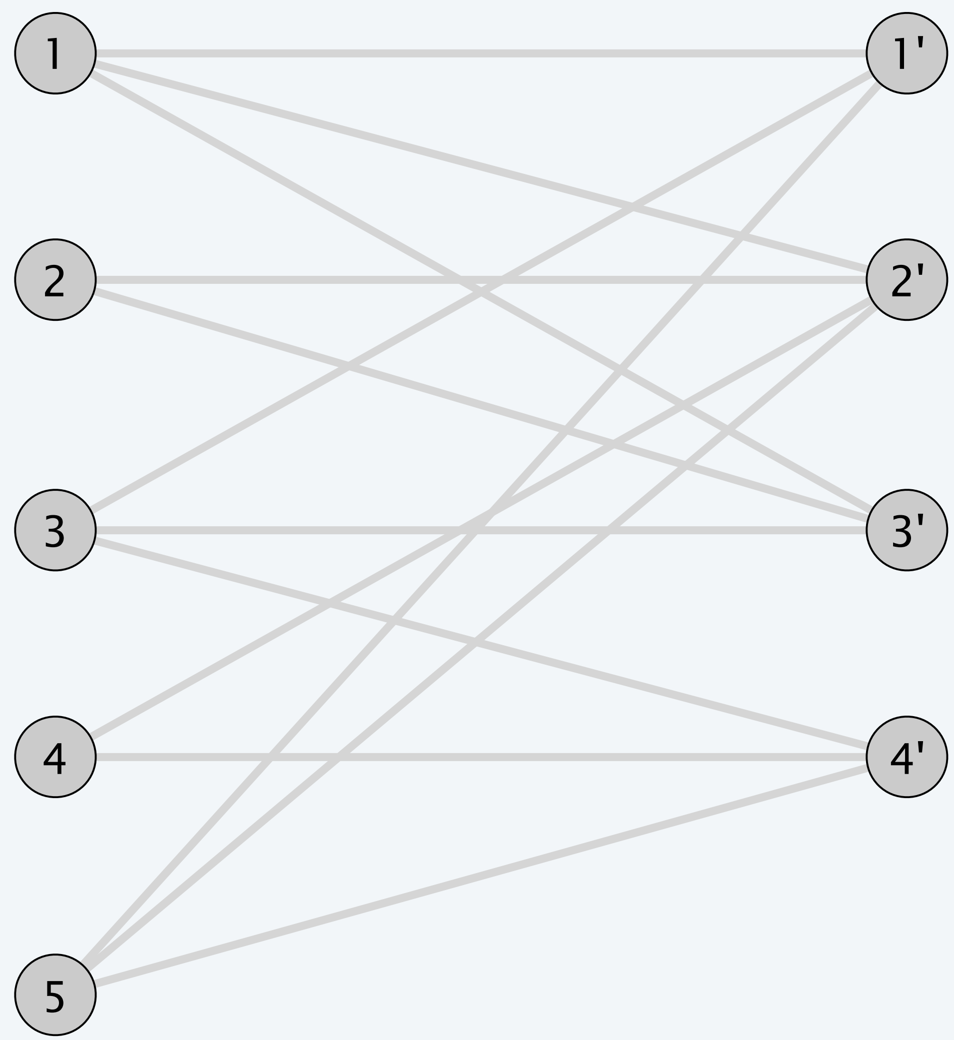
HAM-CYCLE. Given an undirected graph $G = (V, E)$, does there exist a simple cycle Γ that contains every node in V?



yes

Hamilton cycle

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



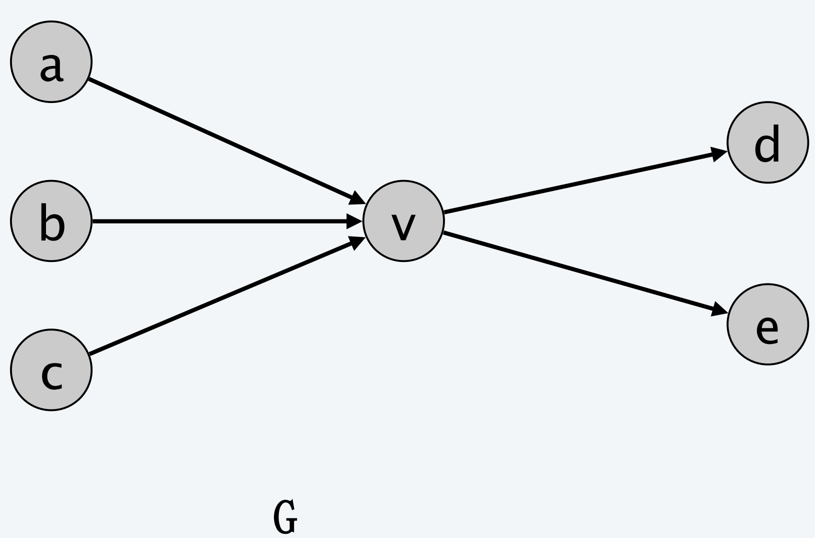
no

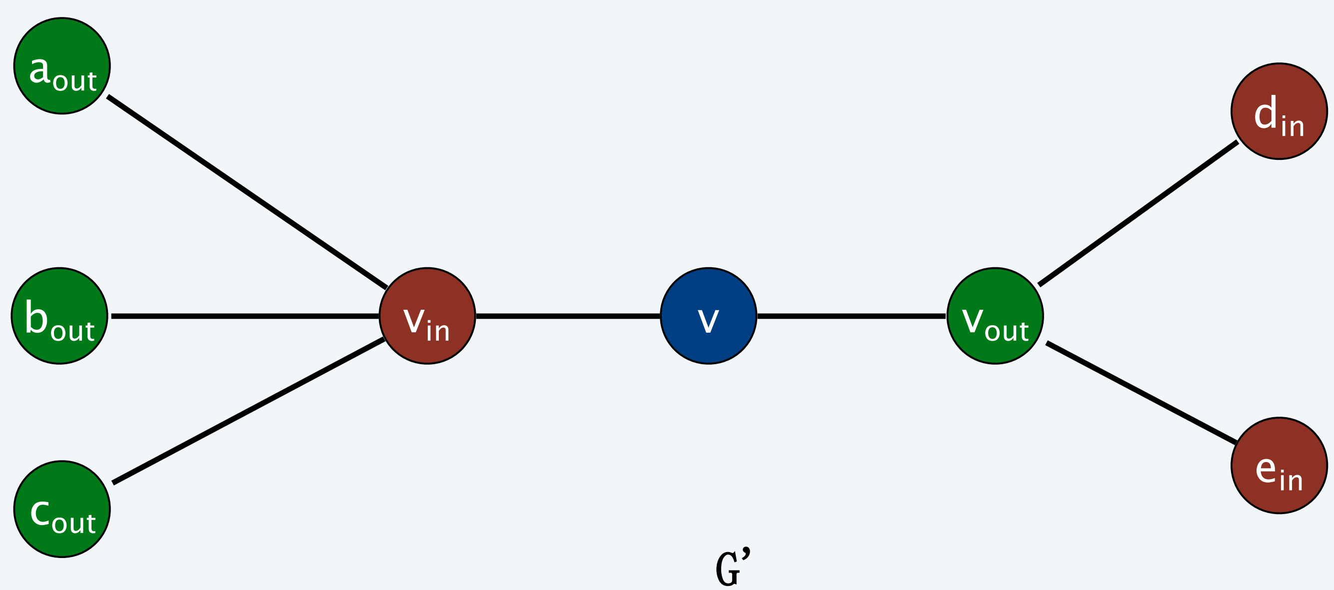
Directed hamilton cycle reduces to hamilton cycle

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

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

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





Directed hamilton cycle reduces to hamilton cycle

Lemma. G has a directed Hamilton cycle iff G' has a Hamilton cycle.

Pf. ®

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

Pf. ∠

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

  $\ldots, B, G, R, B, G, R, B, G, R, B, \ldots$

  $\ldots, B, R, G, B, R, G, B, R, G, B, \ldots$
- Blue nodes in Γ' make up directed Hamilton cycle Γ in G,
  or reverse of one. ▪

3-satisfiability reduces to directed hamilton cycle

Theorem. 3-SAT $\le_P$ DIR-HAM-CYCLE.

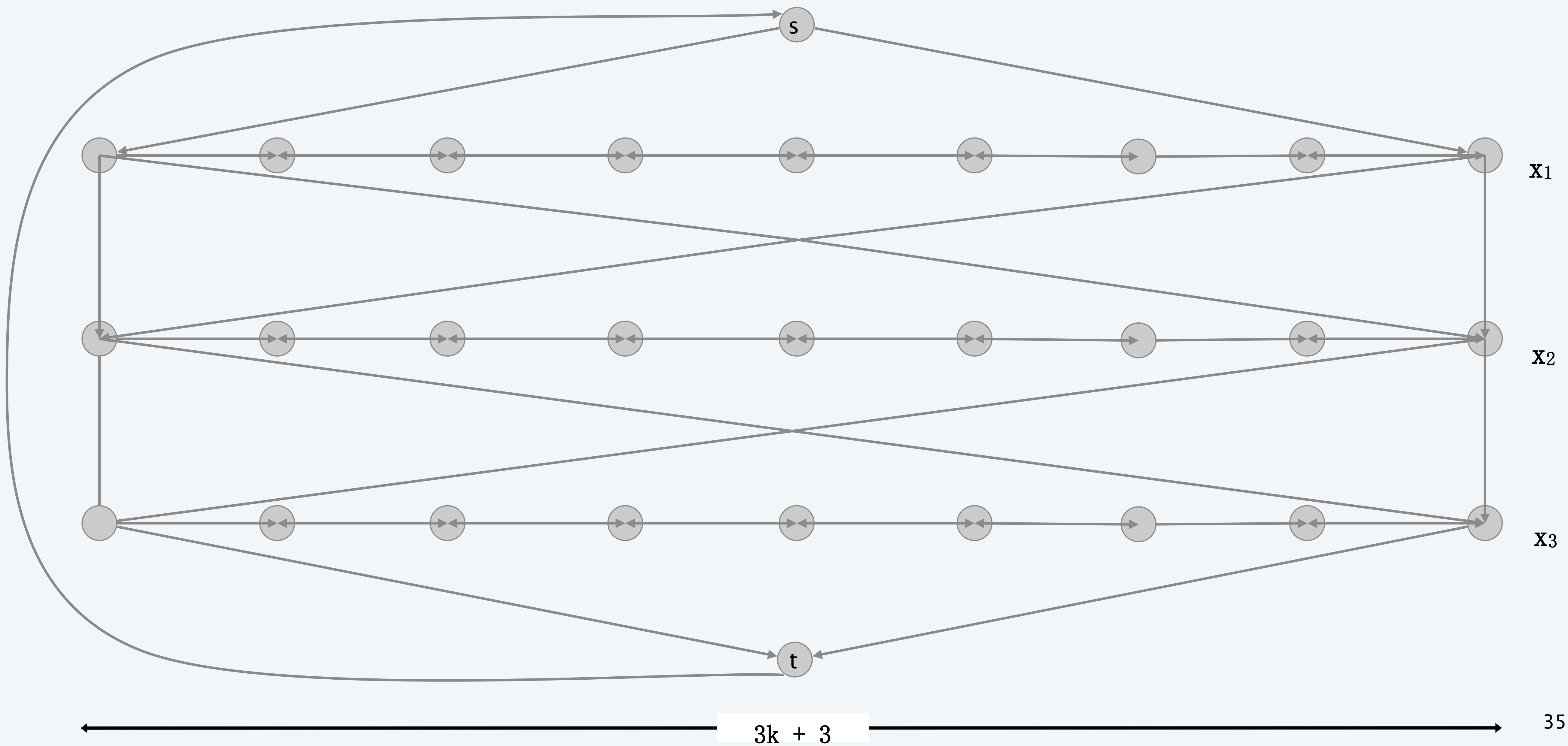
Pf. Given an instance Φ of 3-SAT, we construct an instance of DIR-HAM-CYCLE that has a Hamilton cycle iff Φ is satisfiable.

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

3-satisfiability reduces to directed hamilton cycle

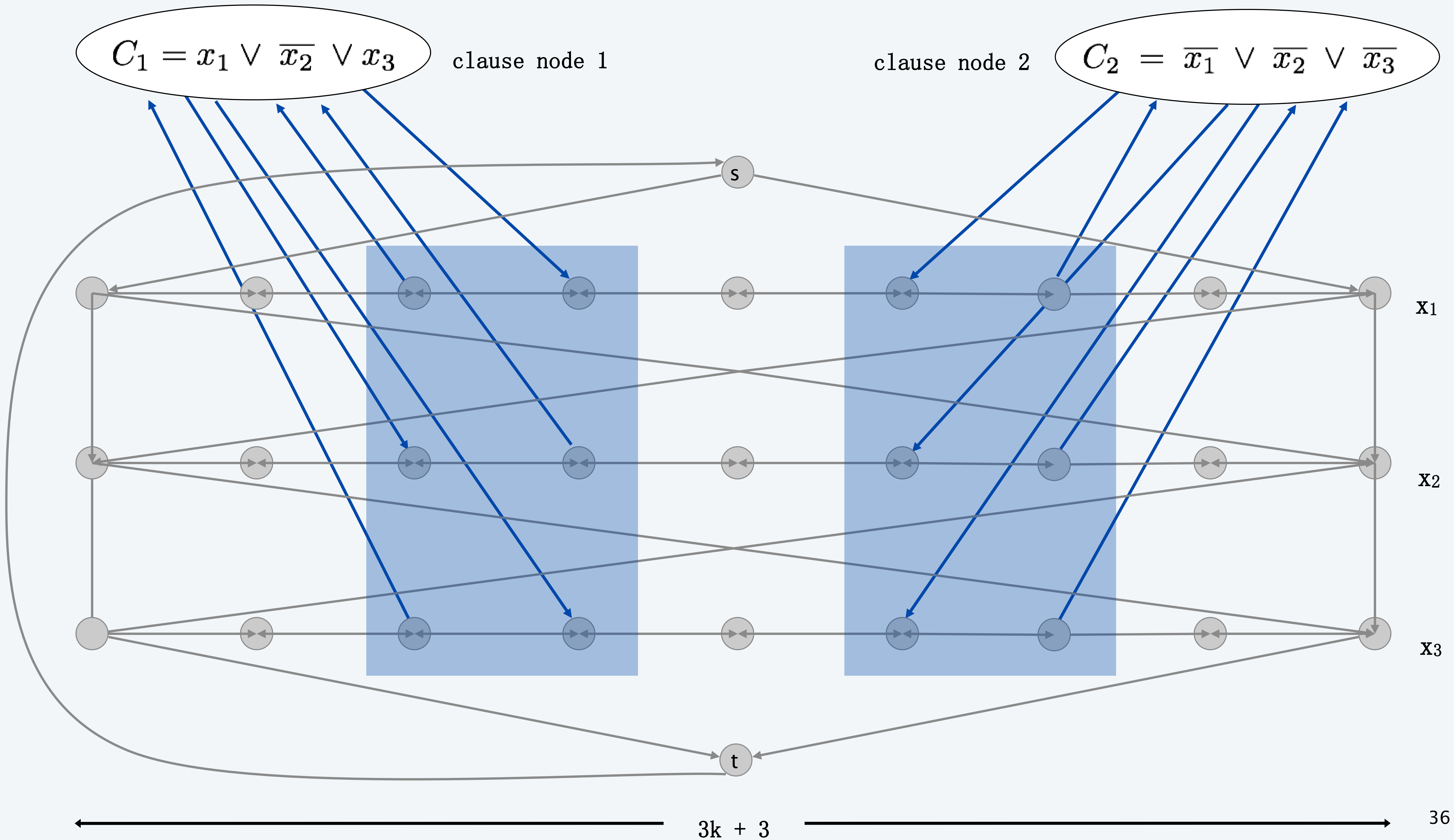
Construction. Given 3-SAT instance Φ with n variables x_i and k clauses.

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



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

- For each clause, add a node and 6 edges.



3-satisfiability reduces to directed hamilton cycle

Lemma. Φ is satisfiable iff G has a Hamilton cycle.

Pf. ®

- Suppose 3-SAT instance has satisfying assignment x^*.
- Then, define Hamilton cycle in G as follows:
  - if $x^*_i = true$, traverse row i from left to right
  - if $x^*_i = false$, 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 clause node C_j into cycle (and we splice in C_j exactly once)

3-satisfiability reduces to directed hamilton cycle

Lemma. Φ is satisfiable iff G has a Hamilton cycle.

Pf. $\angle$

- Suppose G has a Hamilton cycle Γ.
- If Γ enters clause node C_j , it must depart on mate edge.
  - nodes immediately before and after C_j are connected by an edge $e \in E$
  - removing C_j from cycle, and replacing it with edge e yields Hamilton cycle on $G - \{ C_j \}$
- Continuing in this way, we are left with a Hamilton cycle Γ' in $G - \{ C_1 , C_2 , \ldots, C_k \}$.
- Set $x^*_i = true$ 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. ▪

3-satisfiability reduces to longest path

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

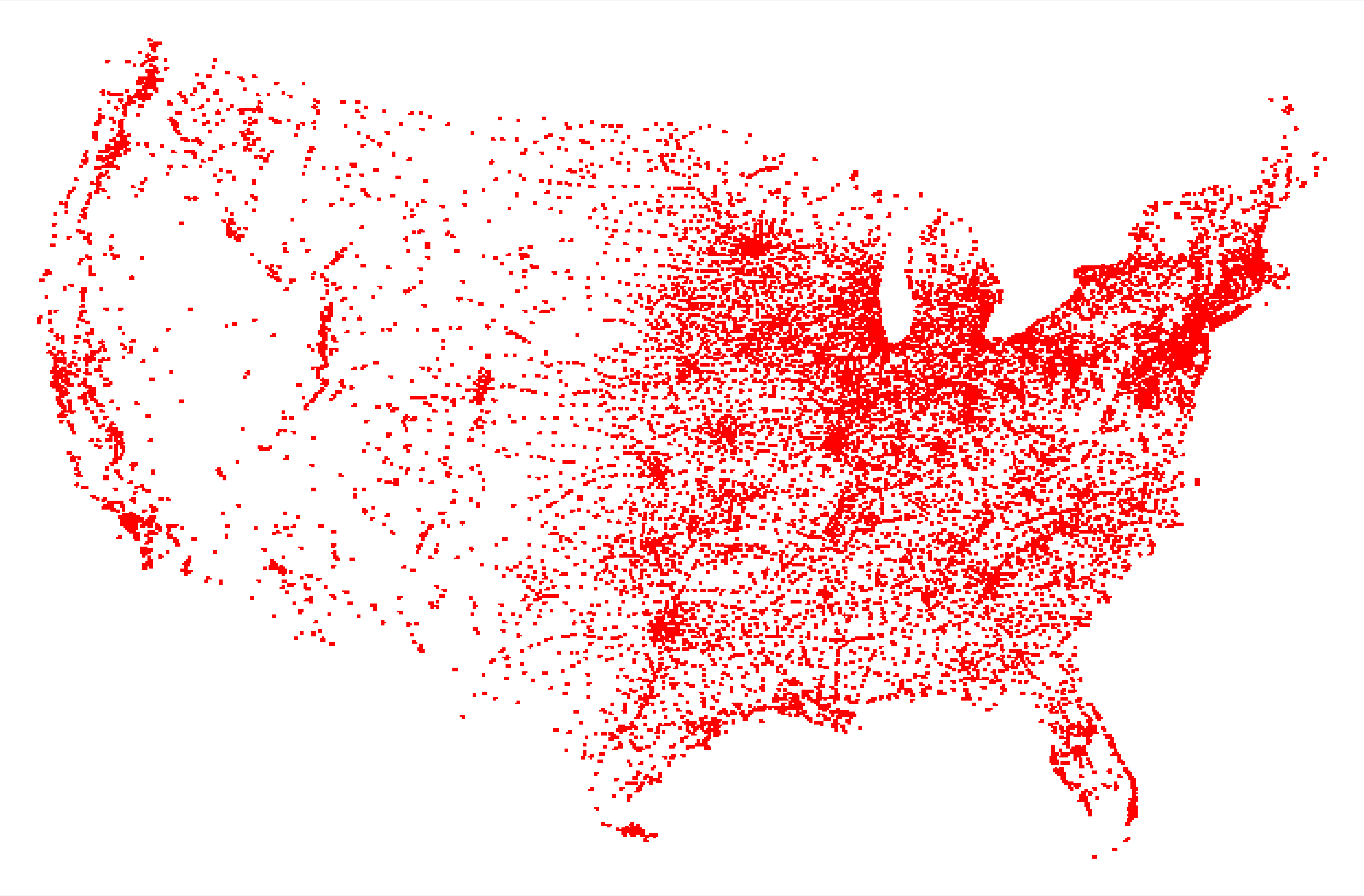
Theorem. 3-SAT $\leq_P$ LONGEST-PATH.

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

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

Traveling salesperson problem

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

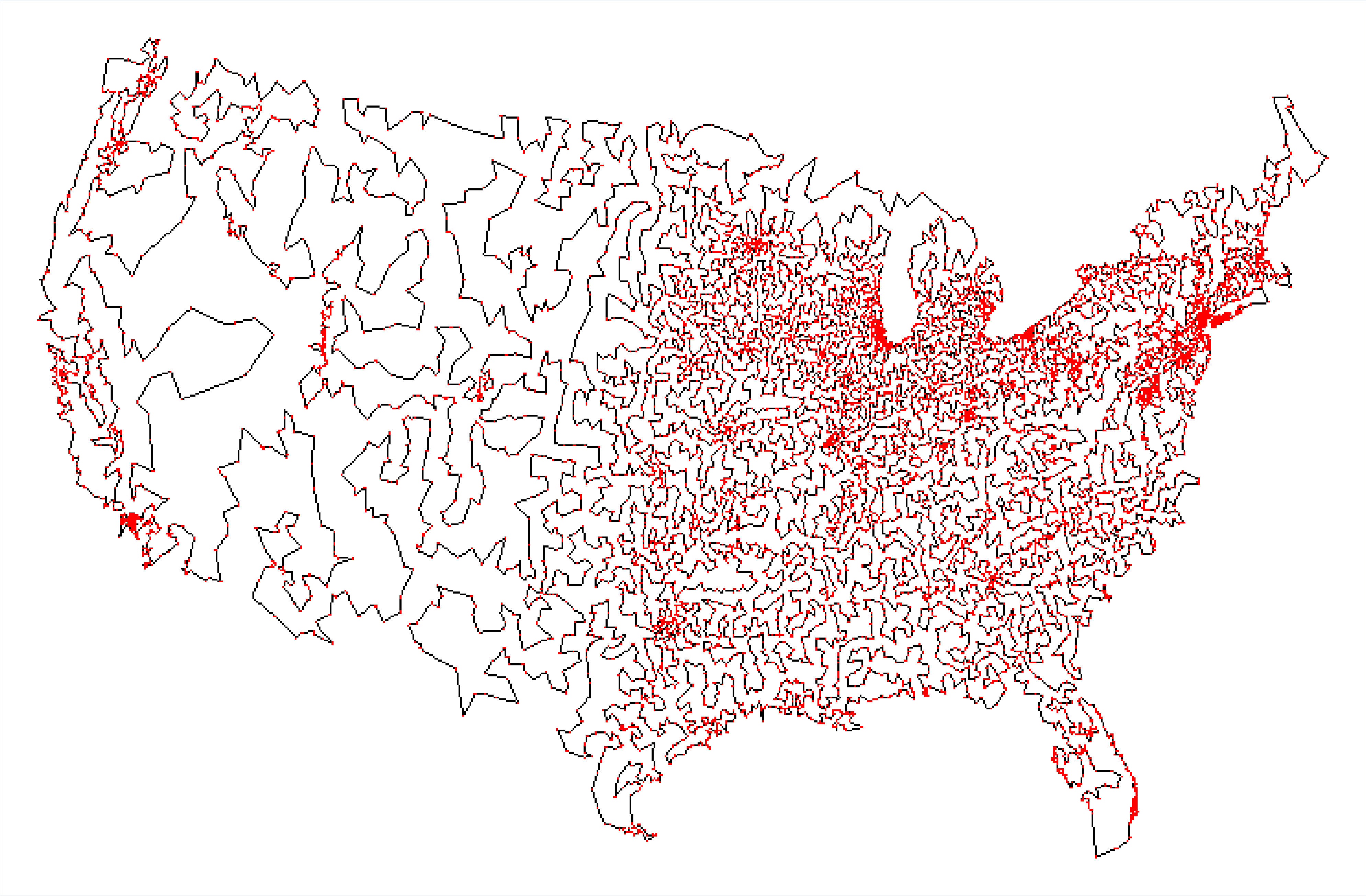


13,509 cities in the United States

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

http://www.tsp.gatech.edu

Traveling salesperson problem

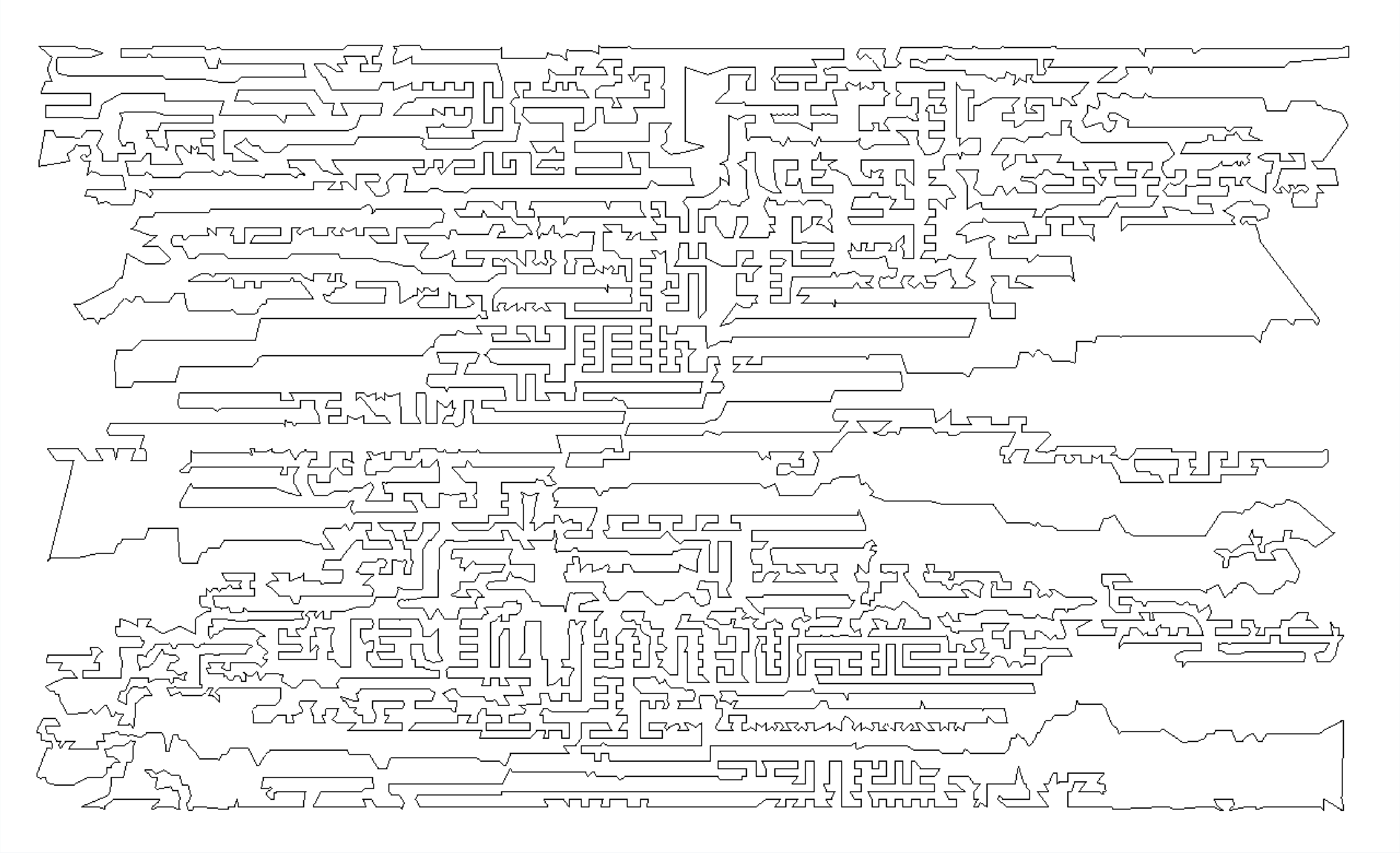
TSP. Given a set of n cities and a pairwise distance function $d(u, v)$, is there a tour of length $\leq D$?

11,849 holes to drill in a programmed logic array

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

http://www.tsp.gatech.edu

Hamilton cycle reduces to 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 an undirected graph $G = (V, E)$, does there exist a simple cycle Γ that contains every node in V?

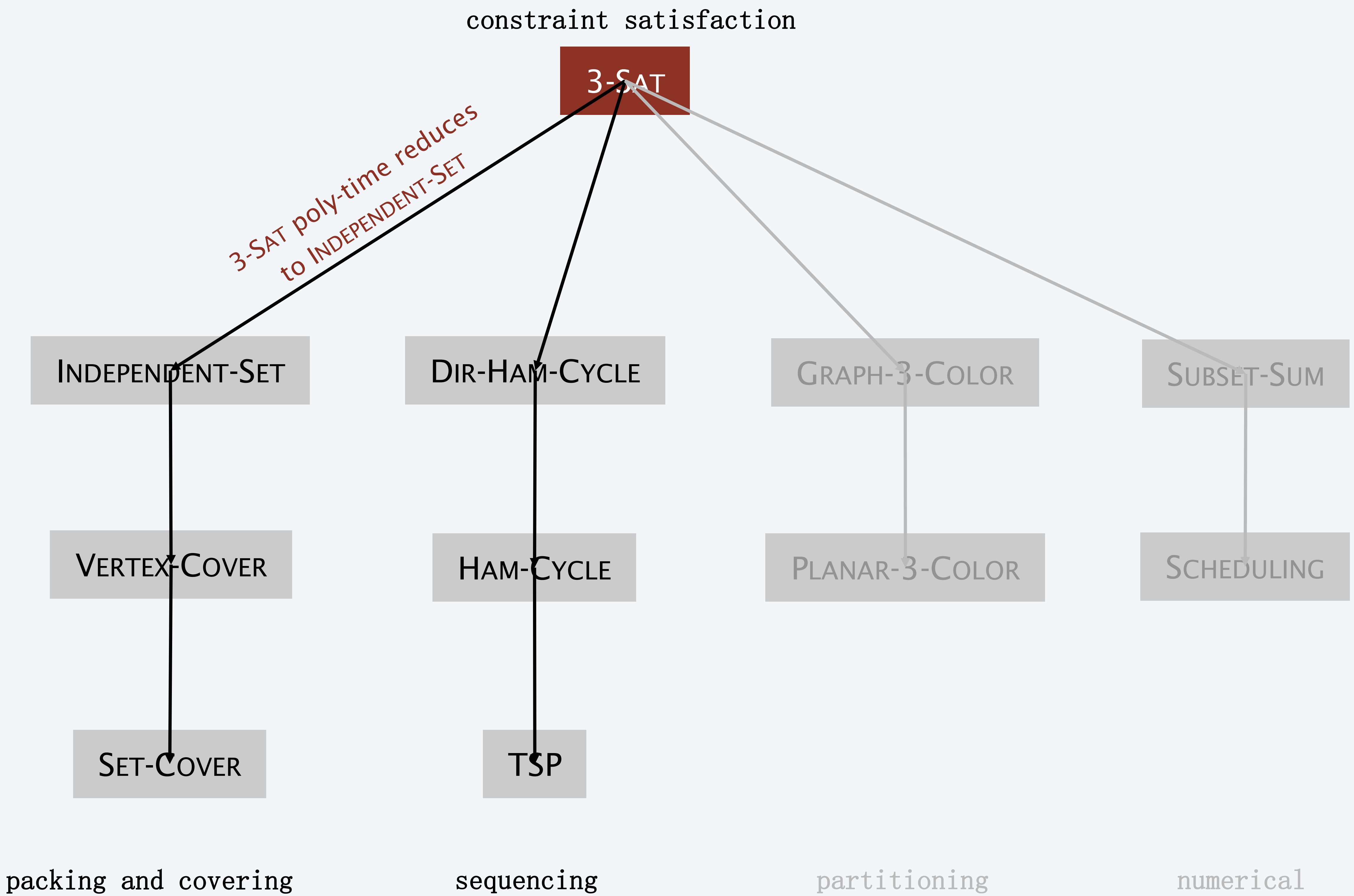
Theorem. HAM-CYCLE $\leq_P$ TSP.

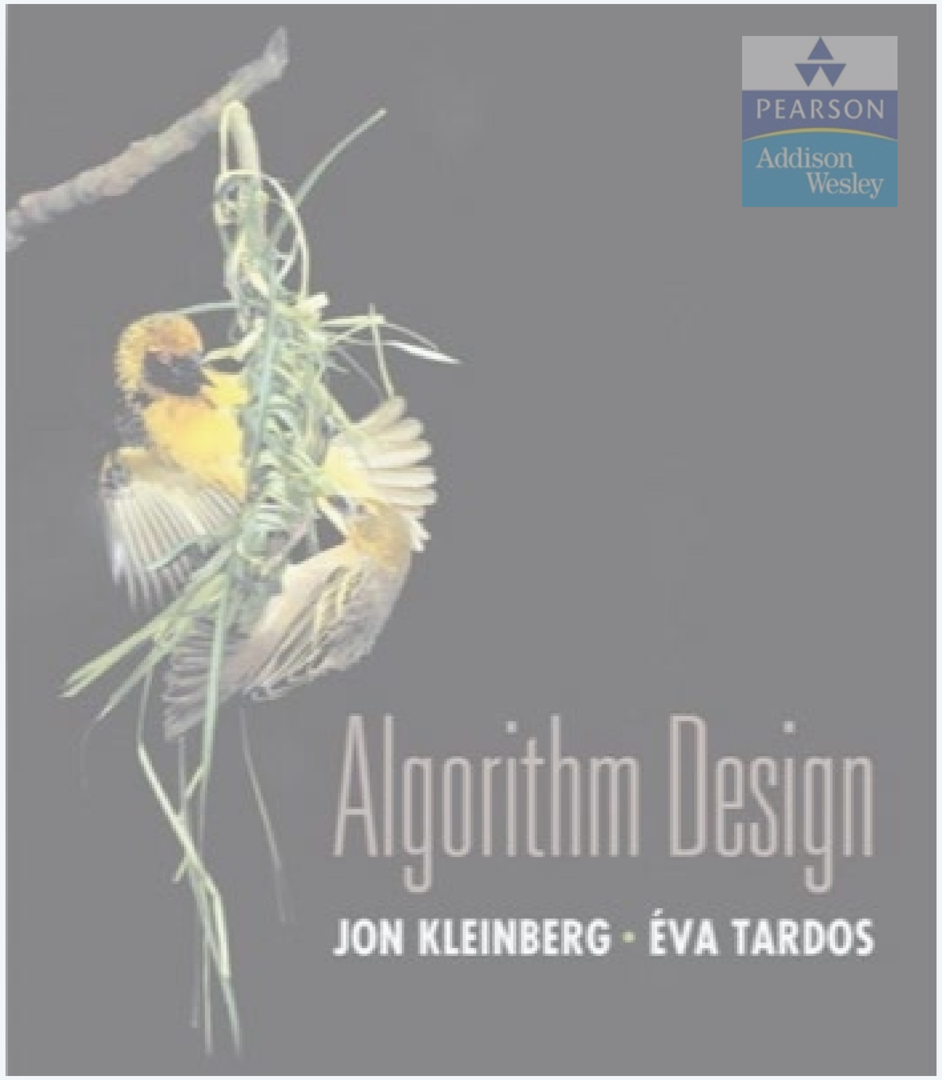
Pf.

- 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 has a Hamilton cycle. ▪

Remark. TSP instance satisfies triangle inequality: $d(u, w) \leq d(u, v) + d(v, w)$.

Polynomial-time reductions





8. INTRACTABILITY I

- *poly-time reductions*
- *packing and covering problems*
- *constraint satisfaction problems*
- *sequencing problems*
- *partitioning problems*
- *graph coloring*
- *numerical problems*

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?

instructor	course	time
Wayne	COS 226	TTh 11–12:20
Wayne	COS 423	MW 11–12:20
Wayne	COS 423	TTh 11–12:20
Tardos	COS 423	TTh 3–4:20
Tardos	COS 523	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 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?

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

$T_1 = \{ x_1, y_1, z_2 \}$, $T_2 = \{ x_1, y_2, z_1 \}$, $T_3 = \{ x_1, y_2, z_2 \}$

$T_4 = \{ x_2, y_2, z_3 \}$, $T_5 = \{ x_2, y_3, z_3 \}$,

$T_7 = \{ x_3, y_1, z_3 \}$, $T_8 = \{ x_3, y_1, z_1 \}$, $T_9 = \{ x_3, y_2, z_1 \}$

an instance of 3d-matching (with n = 3)

Remark. Generalization of bipartite 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?

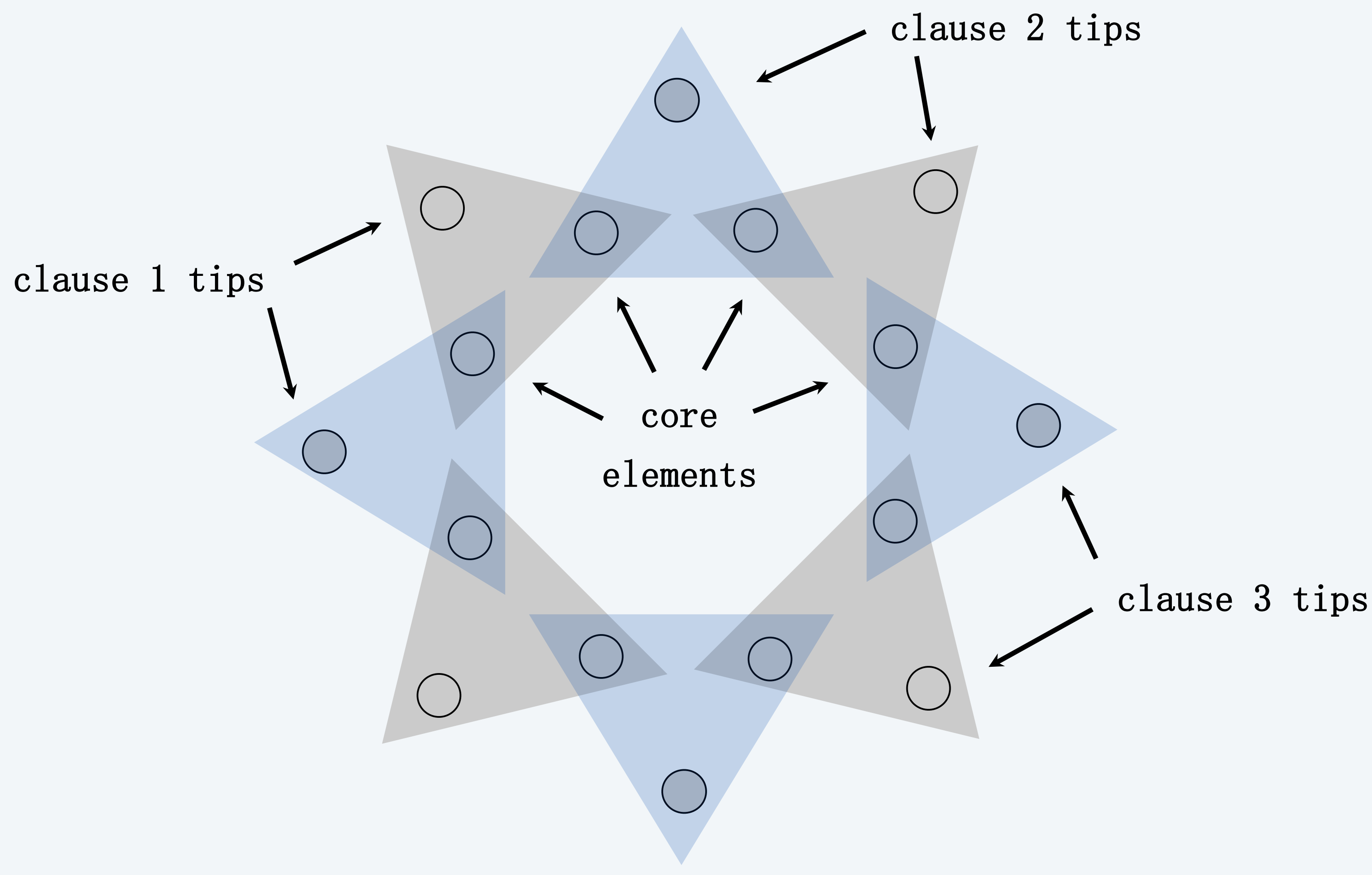
Theorem. 3-SAT $\leq_P$ 3D-MATCHING.

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

Construction. (part 1)

number of clauses

- Create gadget for each variable x_i with $2k$ core elements and $2k$ tip ones.



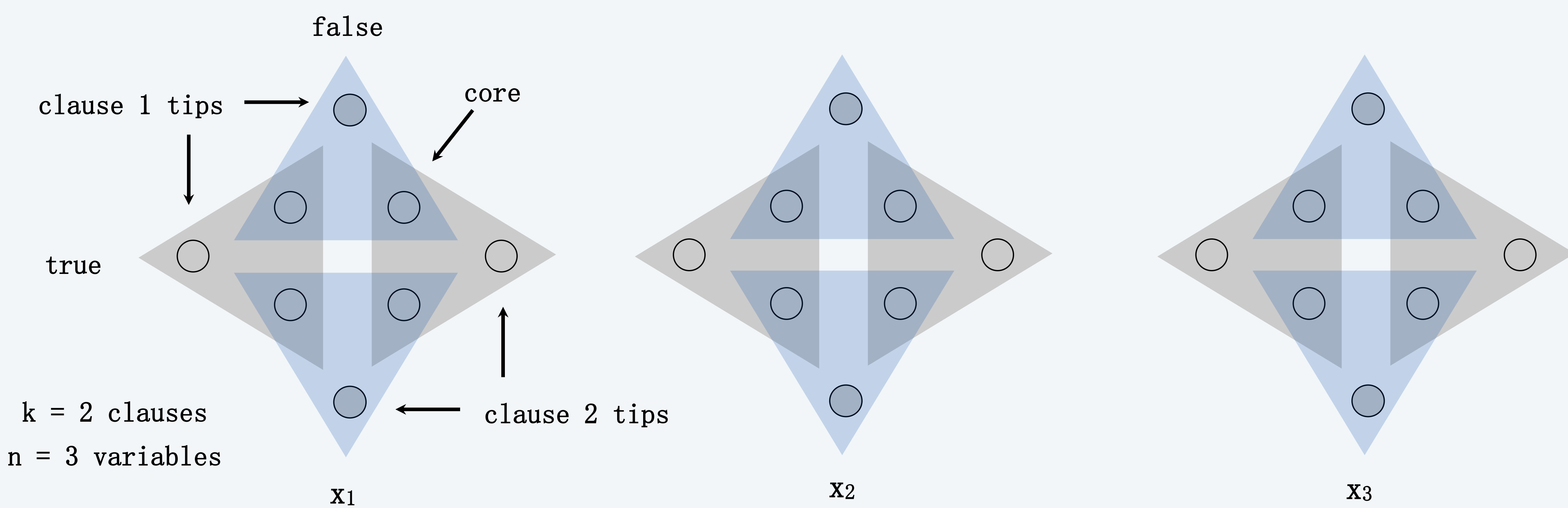
a gadget for variable x_i ($k = 4$)

3-satisfiability reduces to 3-dimensional matching

Construction. (part 1)

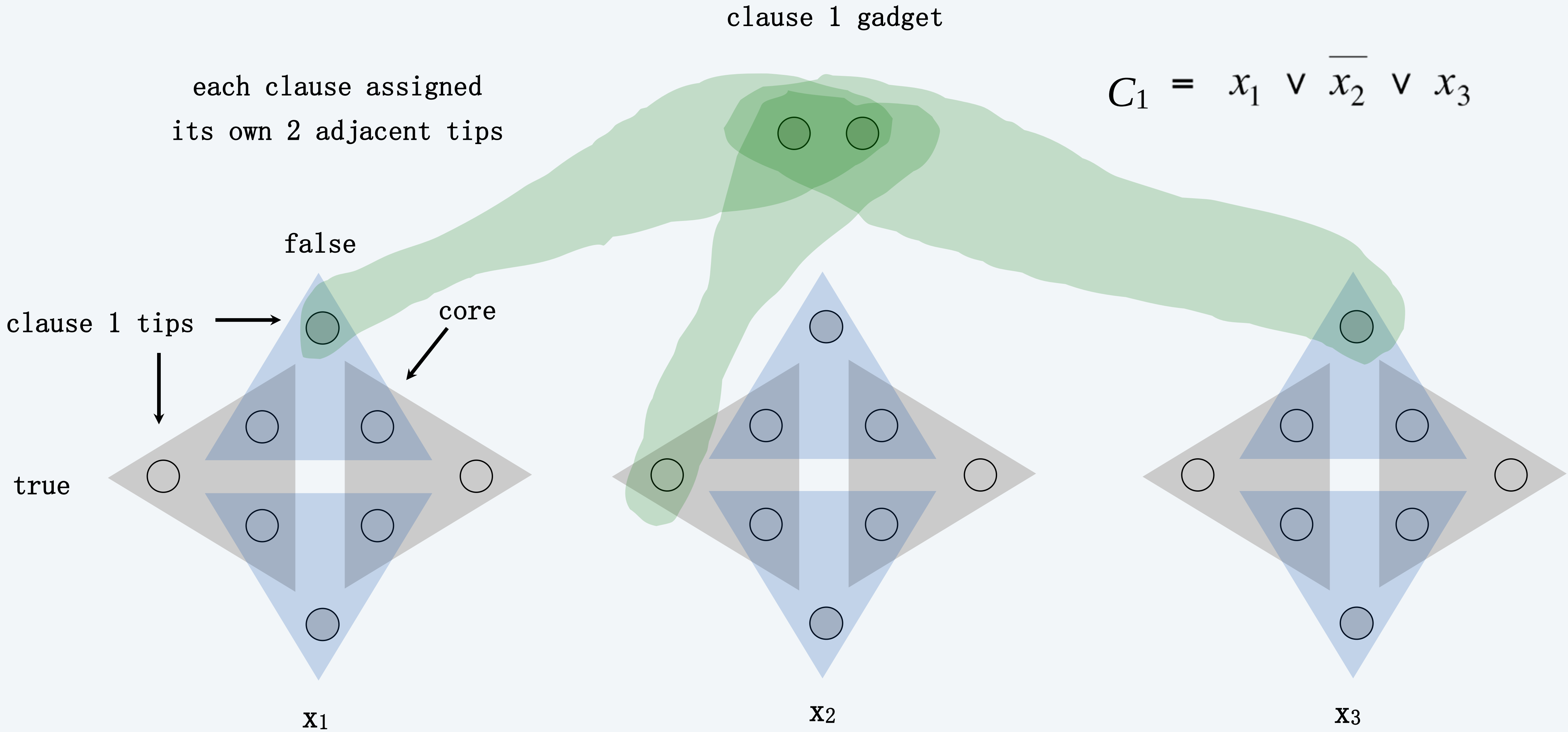
number of clauses

- Create gadget for each variable x_i with $2k$ core elements and $2k$ tip ones.
- No other triples will use core elements.
- In gadget for x_i, any perfect matching must use either all gray triples (corresponding to $x_i = true$) or all blue ones (corresponding to $x_i = false$).



Construction. (part 2)

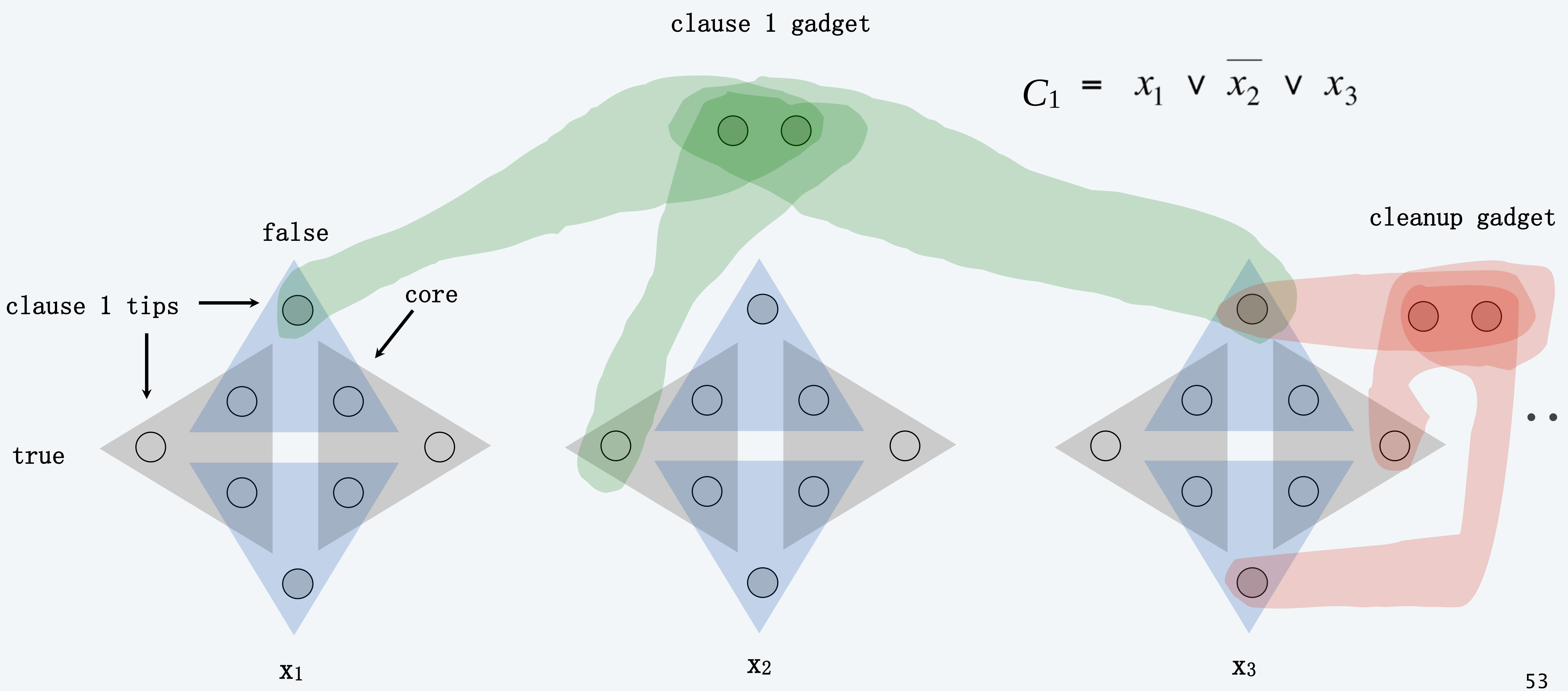
- Create gadget for each clause C_j with two elements and three triples.
- Exactly one of these triples will be used in any 3d-matching.
- Ensures any perfect matching uses either (i) grey core of x_1 or (ii) blue core of x_2 or (iii) grey core of x_3.



3-satisfiability reduces to 3-dimensional matching

Construction. (part 3)

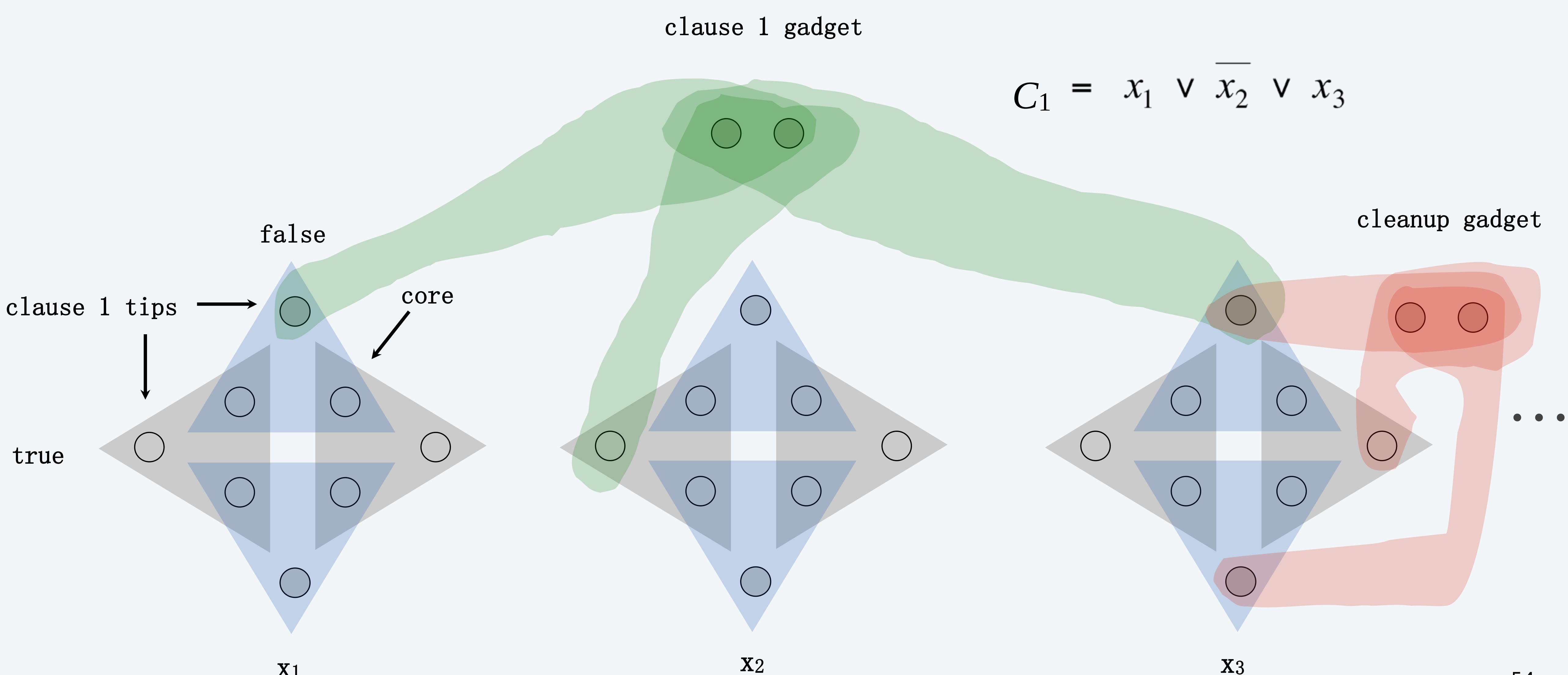
- There are $2nk$ tips: nk covered by blue/gray triples; k by clause triples.
- To cover remaining $(n-1)k$ tips, create $(n-1)k$ cleanup gadgets: same as clause gadget but with $2nk$ triples, connected to every tip.



3-satisfiability reduces to 3-dimensional matching

Lemma. Instance (X, Y, Z) has a perfect matching iff Φ is satisfiable.

Q. What are X, Y, and Z?

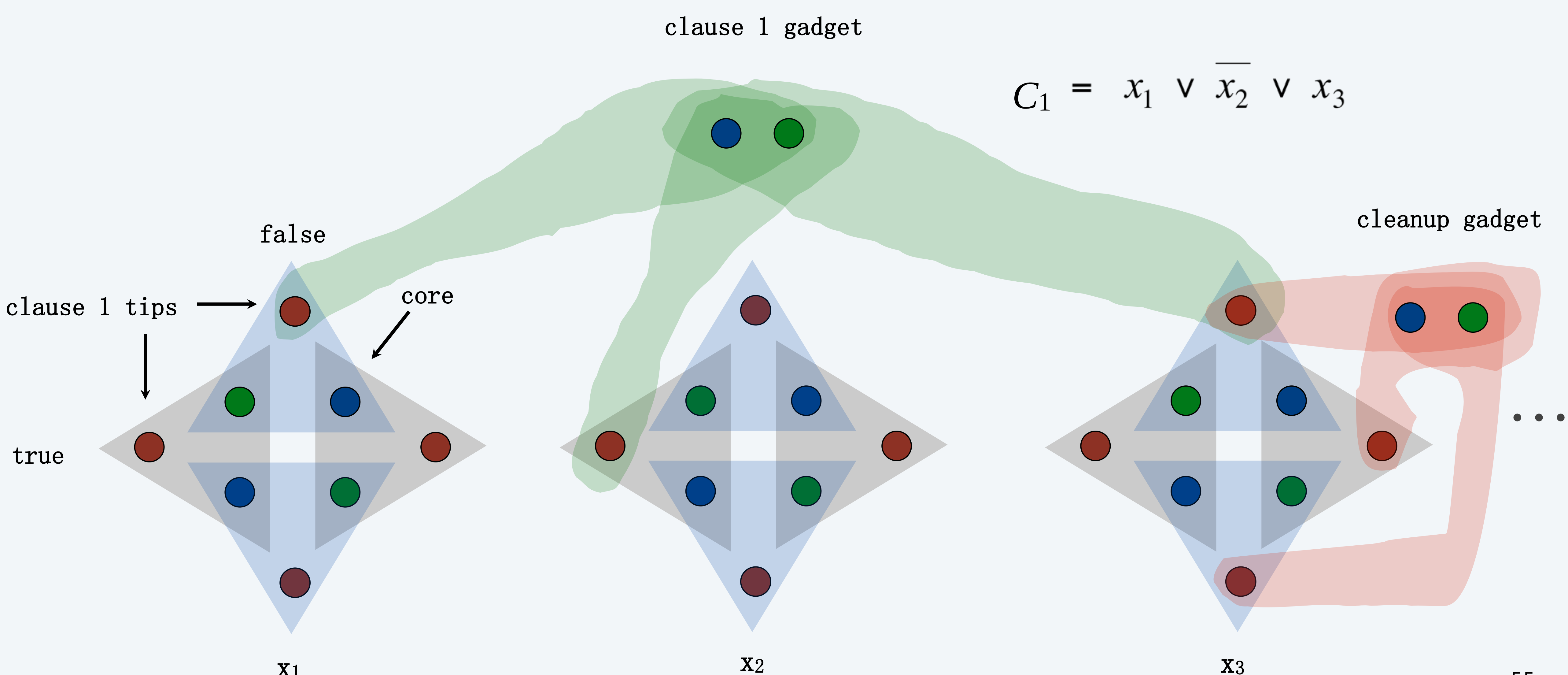


3-satisfiability reduces to 3-dimensional matching

Lemma. Instance (X, Y, Z) has a perfect matching iff Φ is satisfiable.

Q. What are X, Y, and Z?

A. $X = \textit{red}$, $Y = \textit{green}$, and $Z = \textit{blue}$.

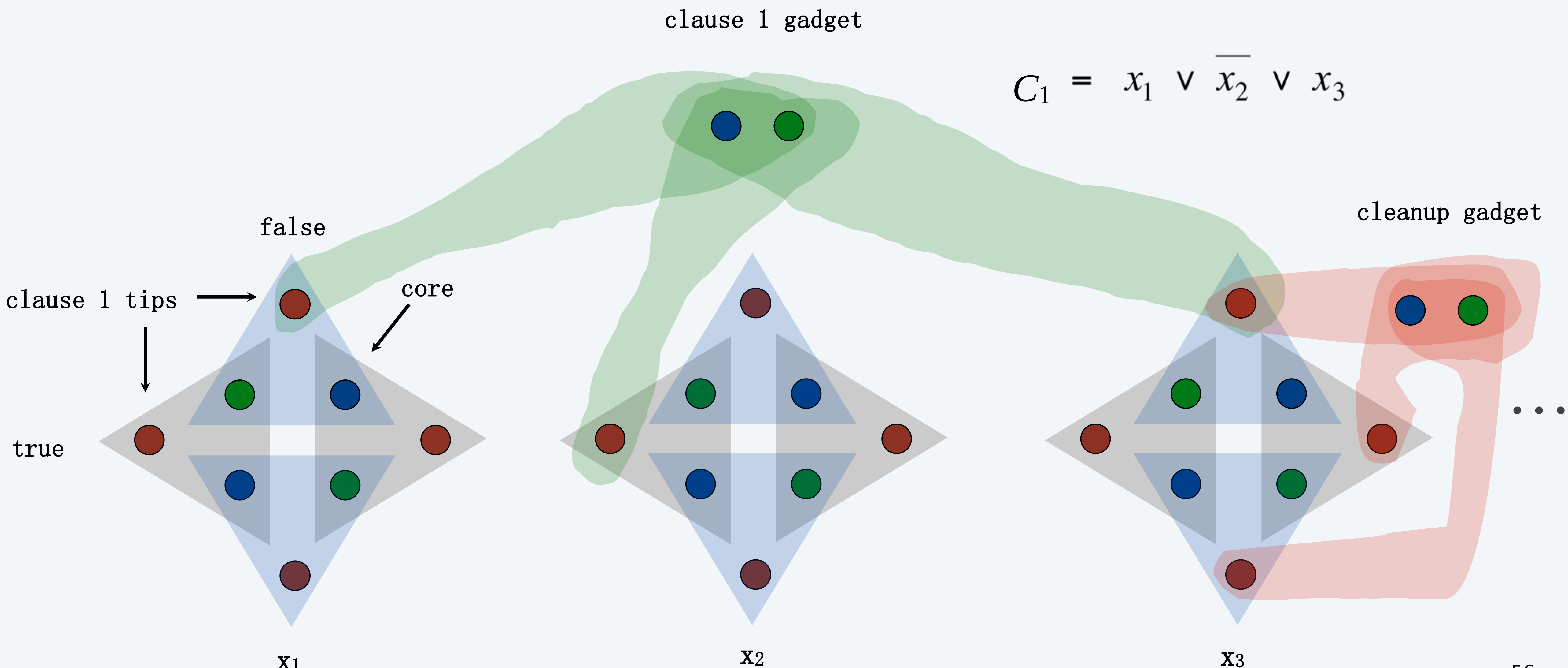


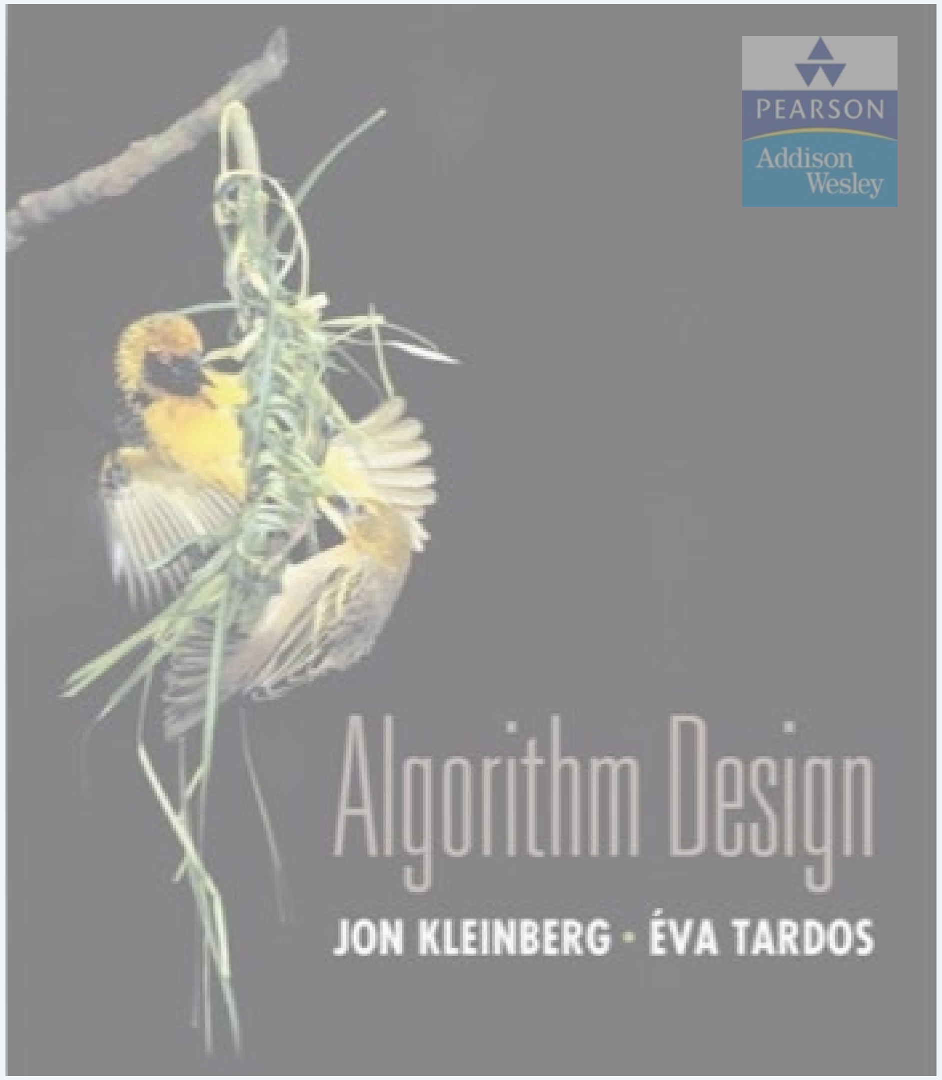
3-satisfiability reduces to 3-dimensional matching

Lemma. Instance (X, Y, Z) has a perfect matching iff Φ is satisfiable.

Pf. ® If 3d-matching, then assign x_i according to gadget x_i.

Pf. ∠ If Φ is satisfiable, use any true literal in C_j to select gadget C_j triple. ▪

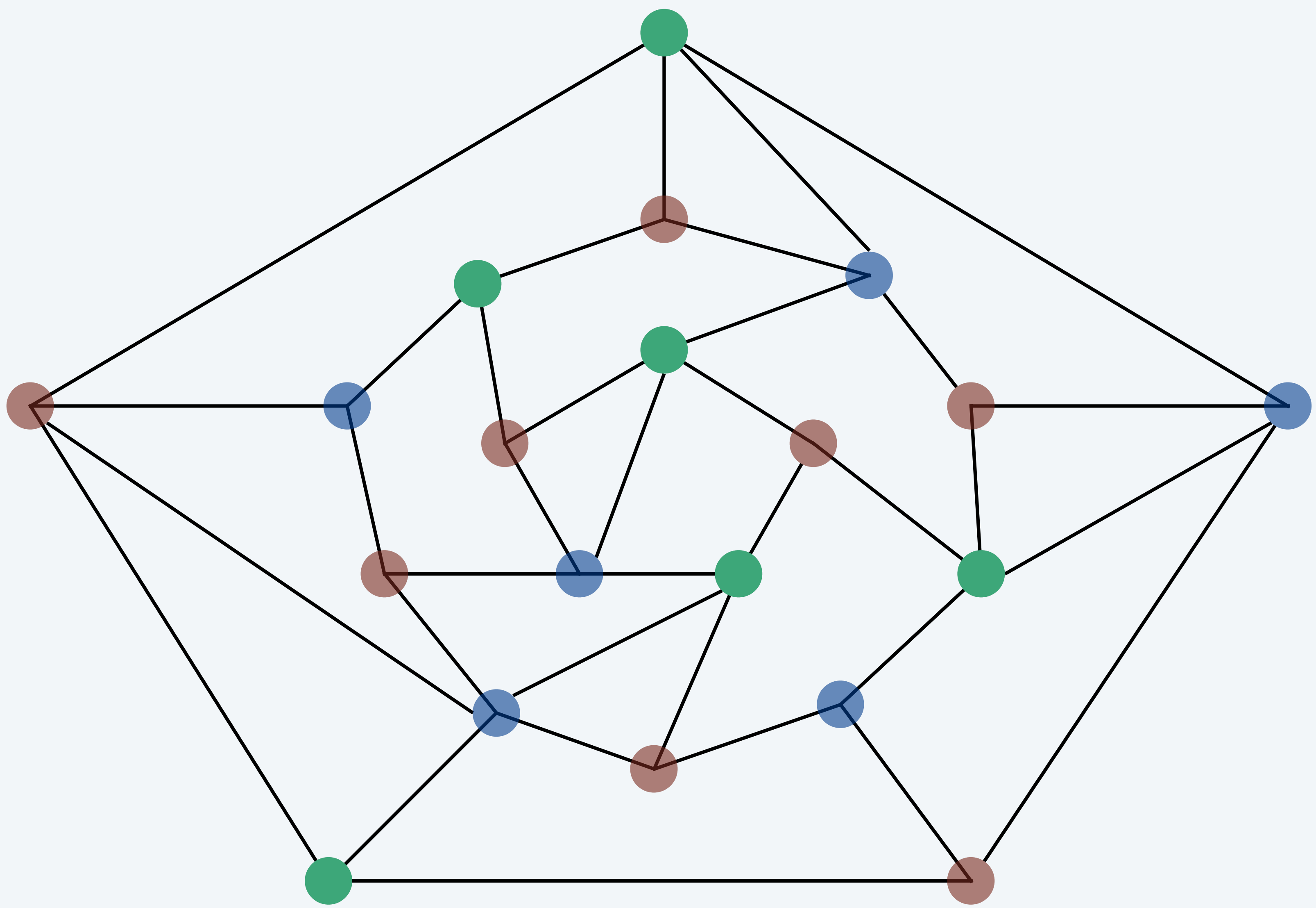




8. INTRACTABILITY I

- *poly-time reductions*
- *packing and covering problems*
- *constraint satisfaction problems*
- *sequencing problems*
- *partitioning problems*
- *graph coloring*
- *numerical problems*

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



yes instance

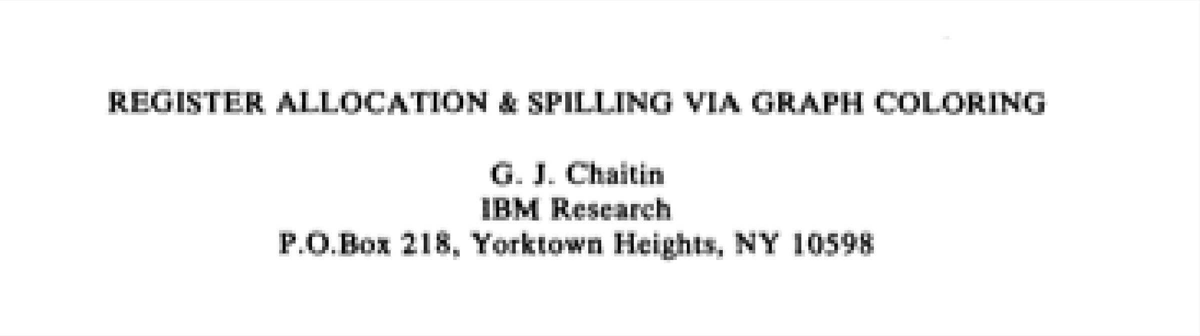
Application: register allocation

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 $\le_P$ K-REGISTER-ALLOCATION for any constant $k \ge 3$.


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

3-satisfiability reduces to 3-colorability

Construction.

(i) Create a graph G with a node for each literal.

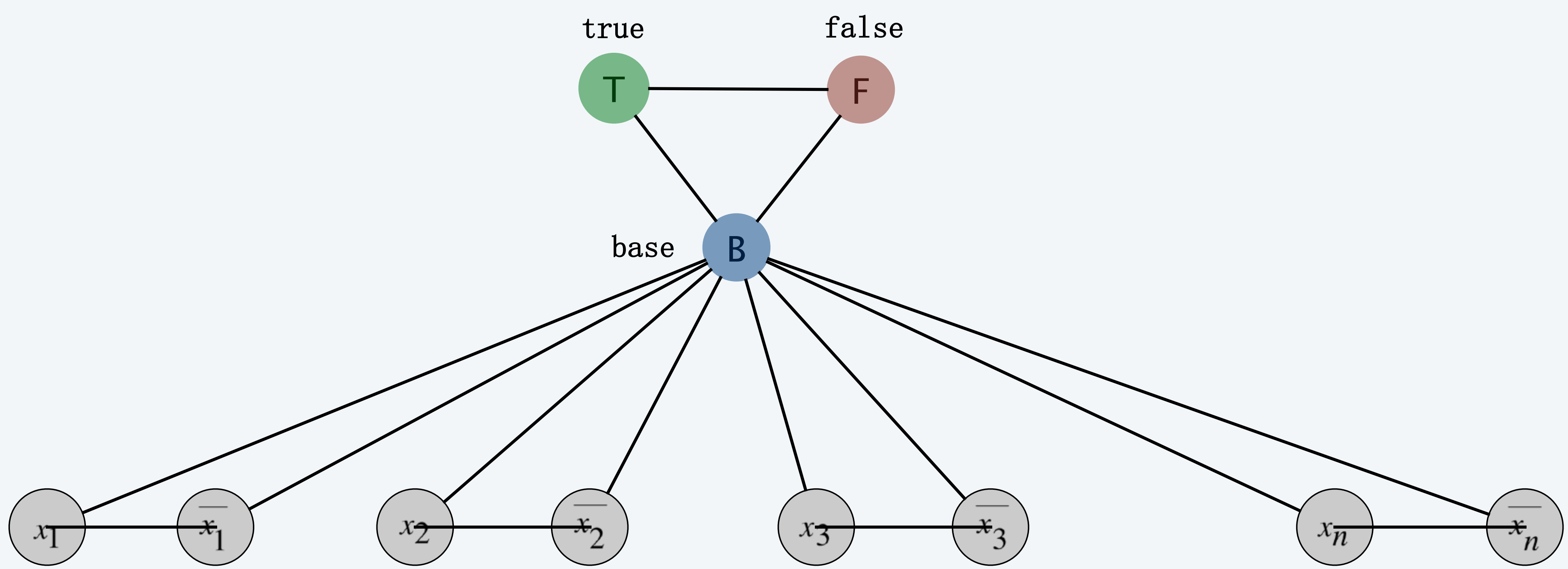
(ii) Connect each literal to its negation.

(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_j, add a gadget of 6 nodes and 13 edges.

to be described later

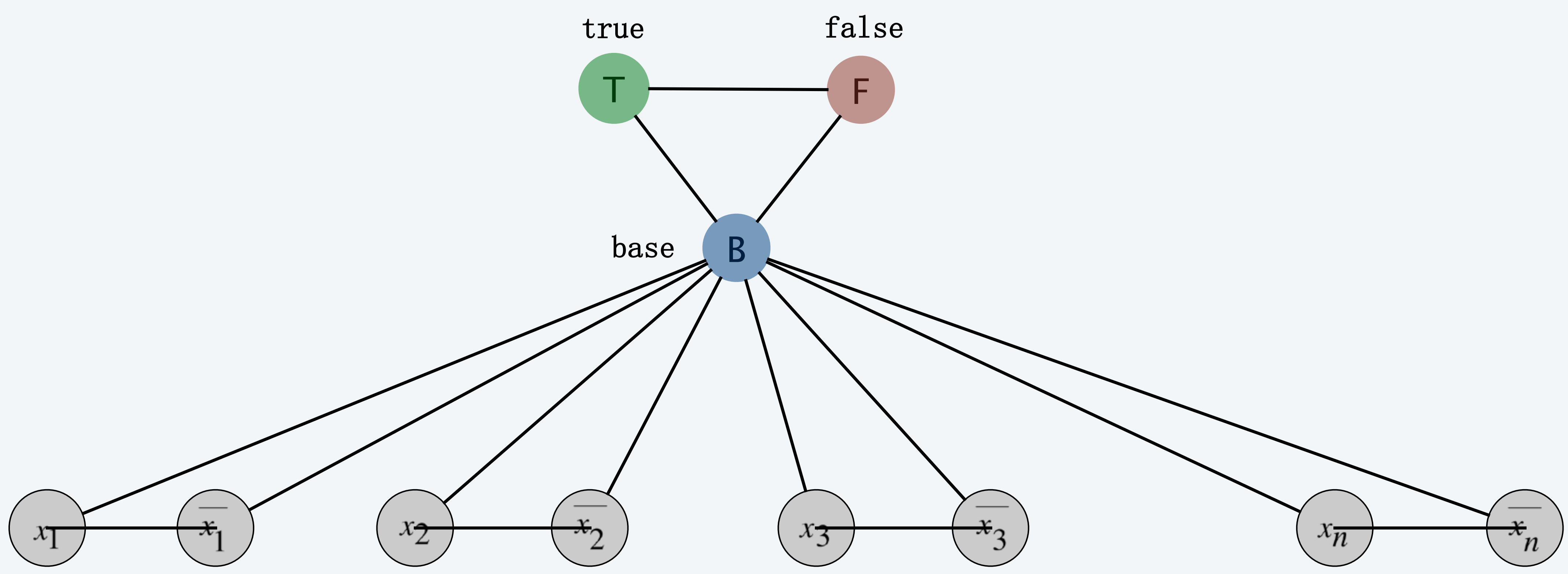


3-satisfiability reduces to 3-colorability

Lemma. Graph G is 3-colorable iff Φ is satisfiable.

Pf. ® Suppose graph G is 3-colorable.

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

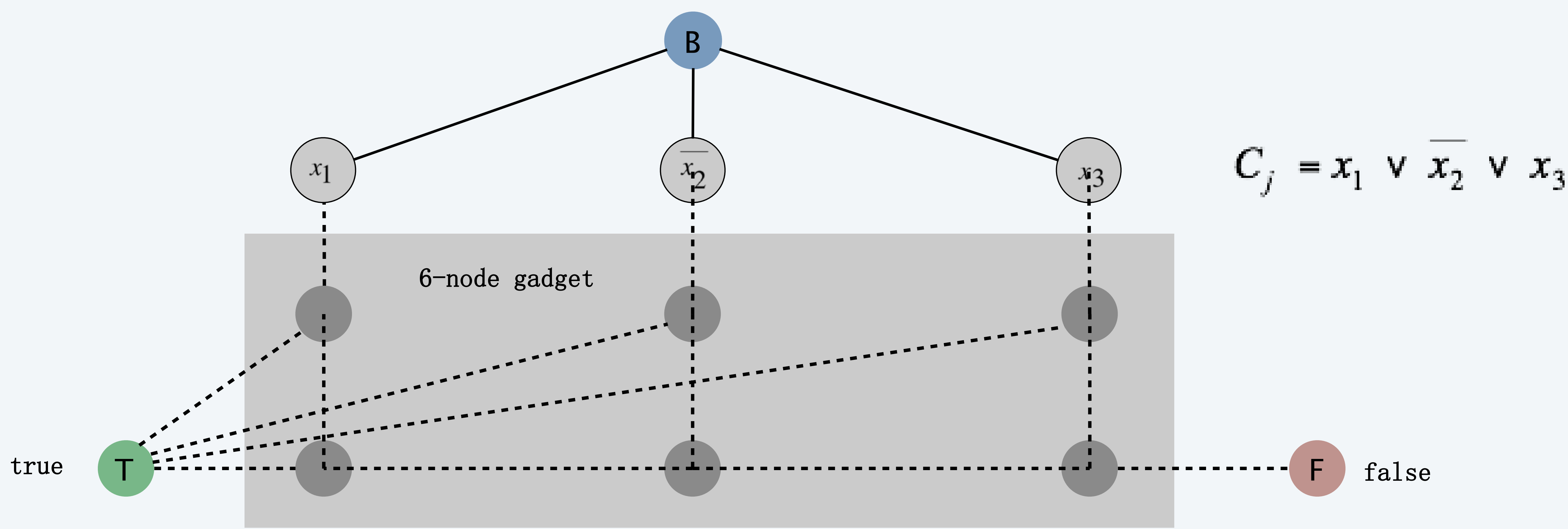


3-satisfiability reduces to 3-colorability

Lemma. Graph G is 3-colorable iff Φ is satisfiable.

Pf. ® Suppose graph G is 3-colorable.

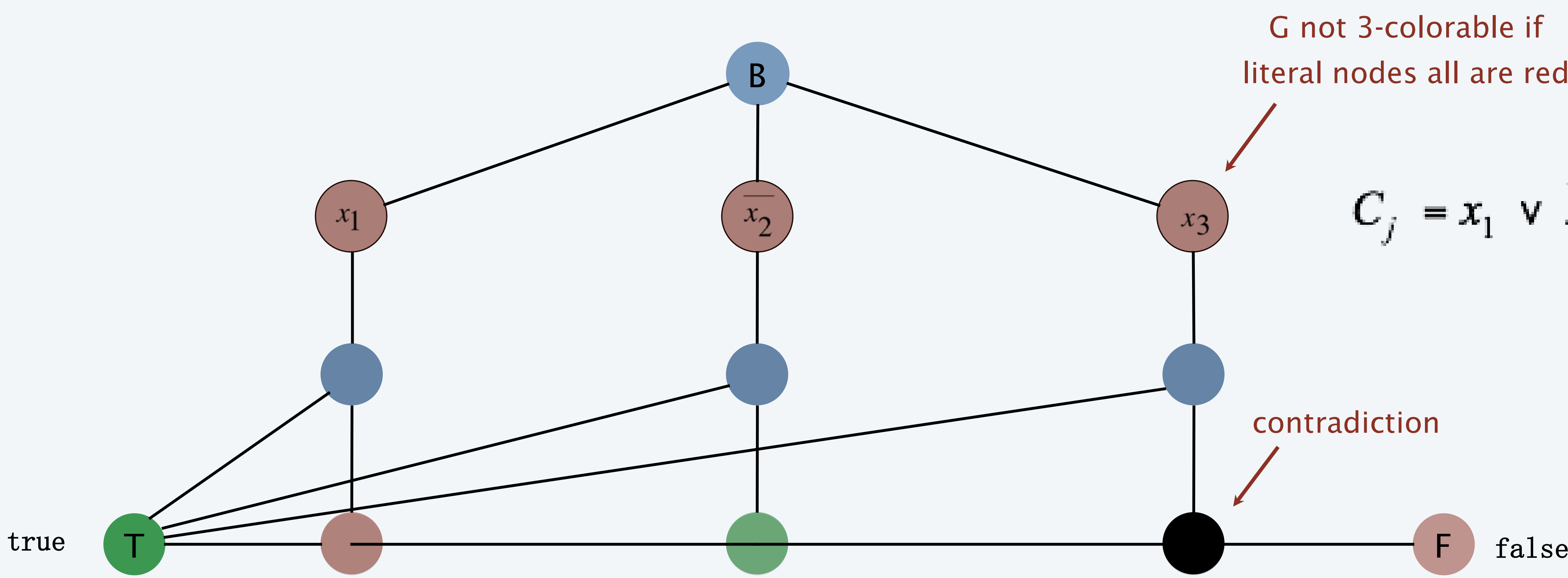
- Consider assignment that sets all T literals to true.
- (iv) ensures each literal is T or F.
- (ii) ensures a literal and its negation are opposites.
- (v) ensures at least one literal in each clause is T.



Lemma. Graph G is 3-colorable iff Φ is satisfiable.

Pf. ® Suppose graph G is 3-colorable.

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

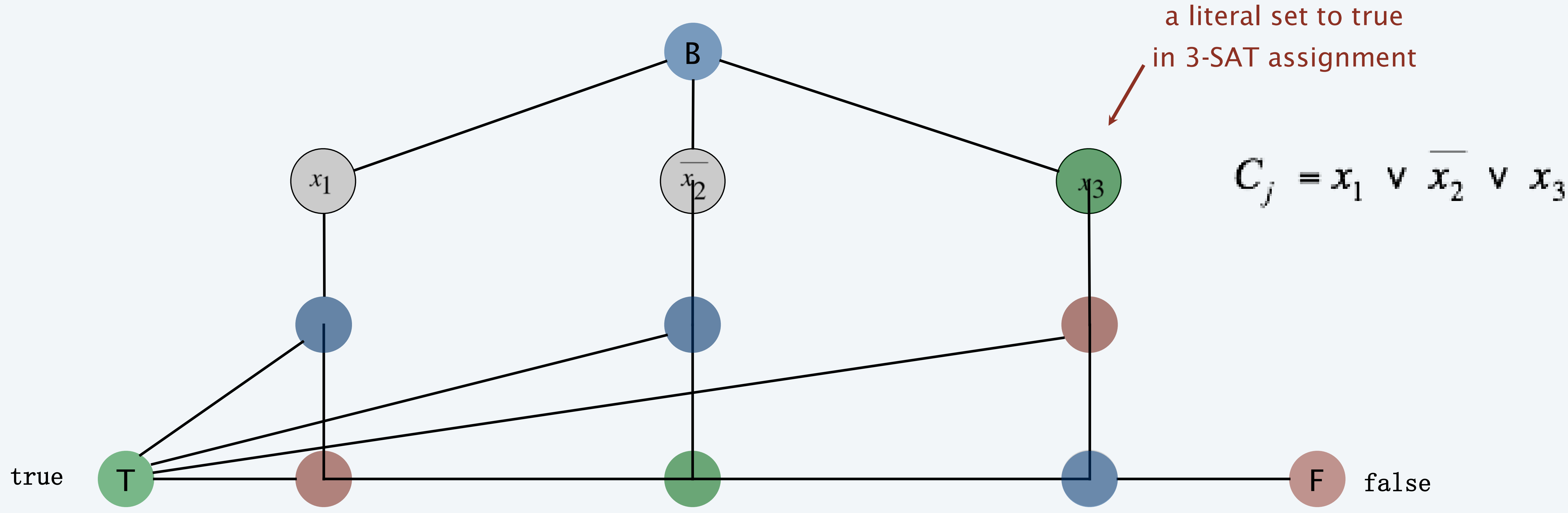


$$C_j = x_1 \vee \overline{x_2} \vee x_3$$

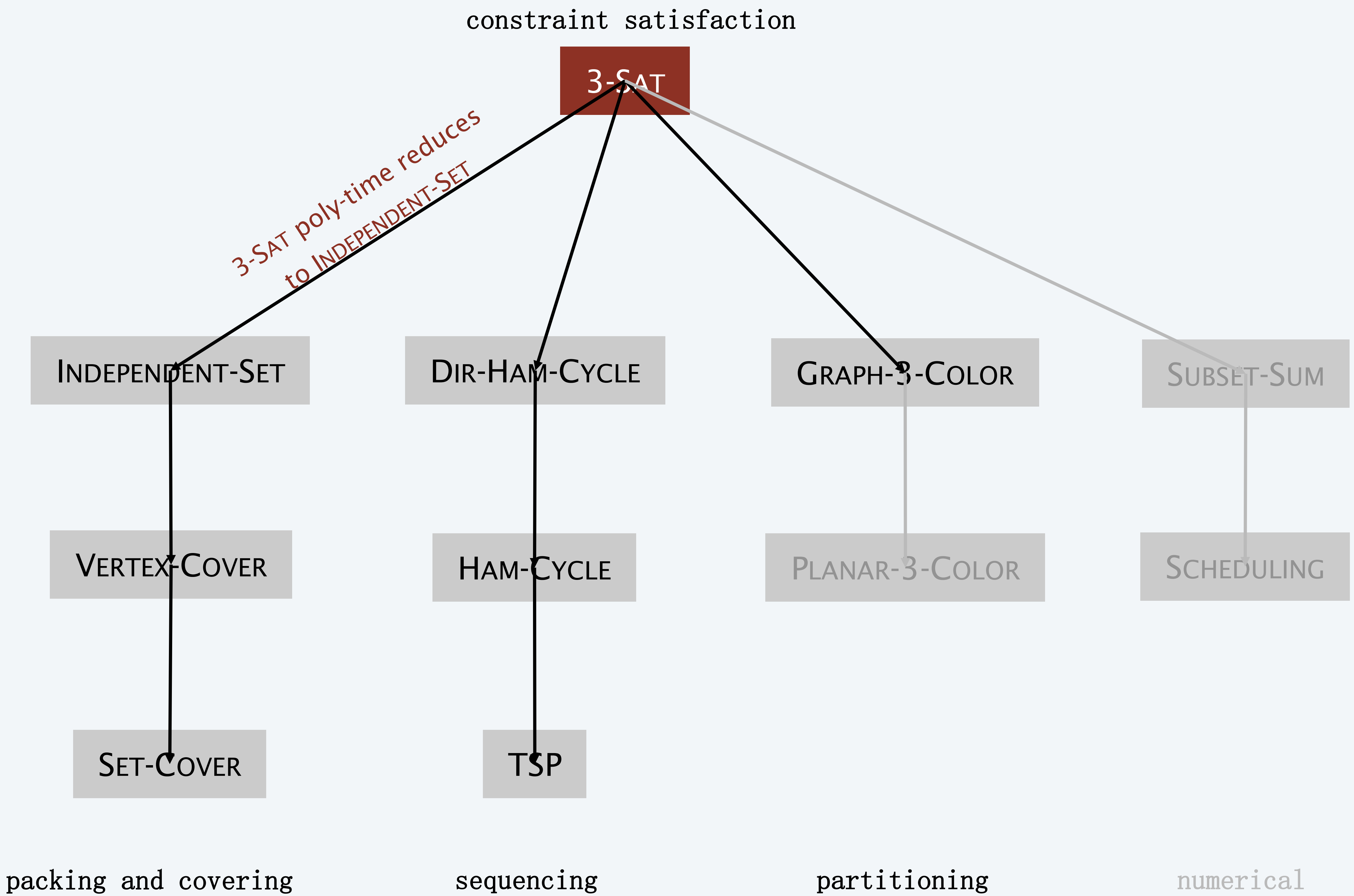
Lemma. Graph G is 3-colorable iff Φ is satisfiable.

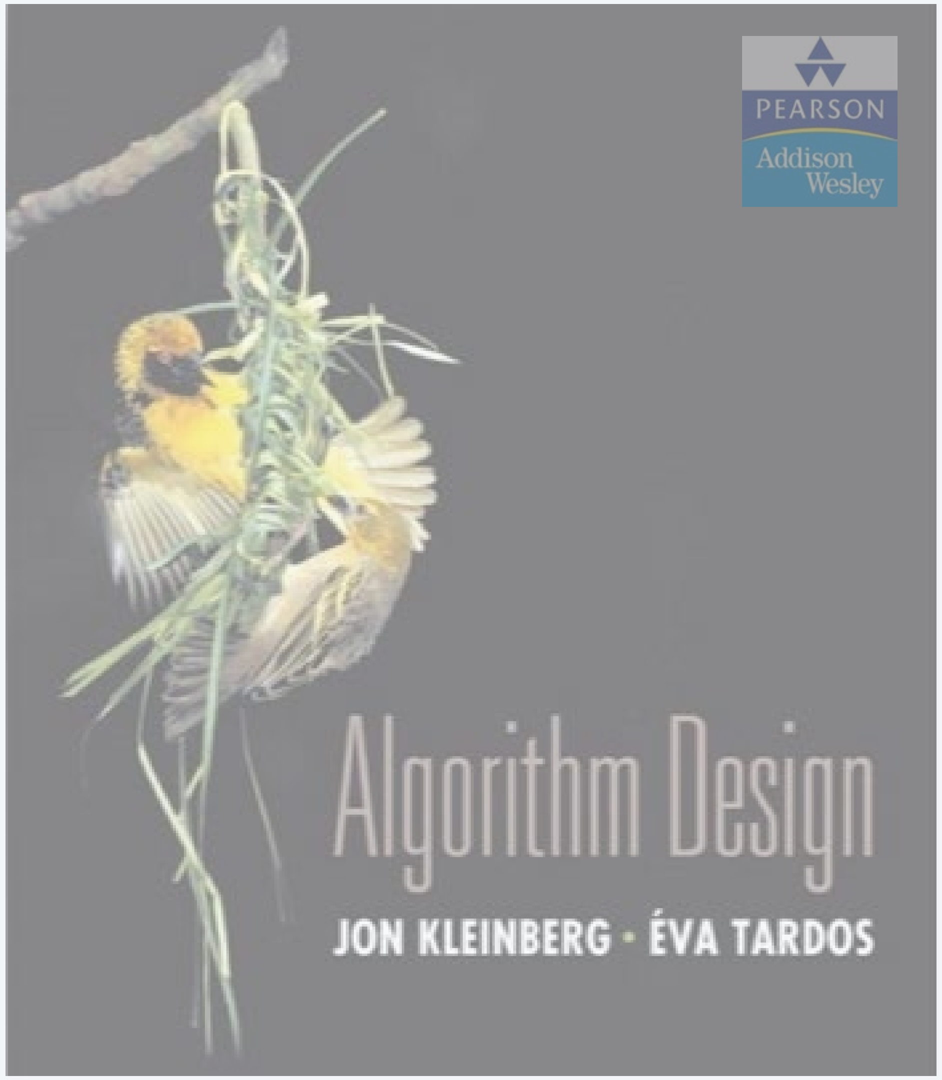
Pf. ⇐ Suppose 3-SAT instance Φ is satisfiable.

- 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. ▪



Polynomial-time reductions





8. INTRACTABILITY I

- *poly-time reductions*
- *packing and covering problems*
- *constraint satisfaction problems*
- *sequencing problems*
- *partitioning problems*
- *graph coloring*
- *numerical problems*

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.

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

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

Construction. Given 3-SAT instance Φ 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.

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

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
	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

3-satisfiability reduces to subset sum

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

Pf. $\Rightarrow$ Suppose Φ is satisfiable.

- Choose integers corresponding to each *true* literal.
- Since Φ 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.

	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
	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

dummies to get clause columns to sum to 4

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

3-SAT instance

3-satisfiability reduces to subset sum

Lemma. Φ 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. ▪

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

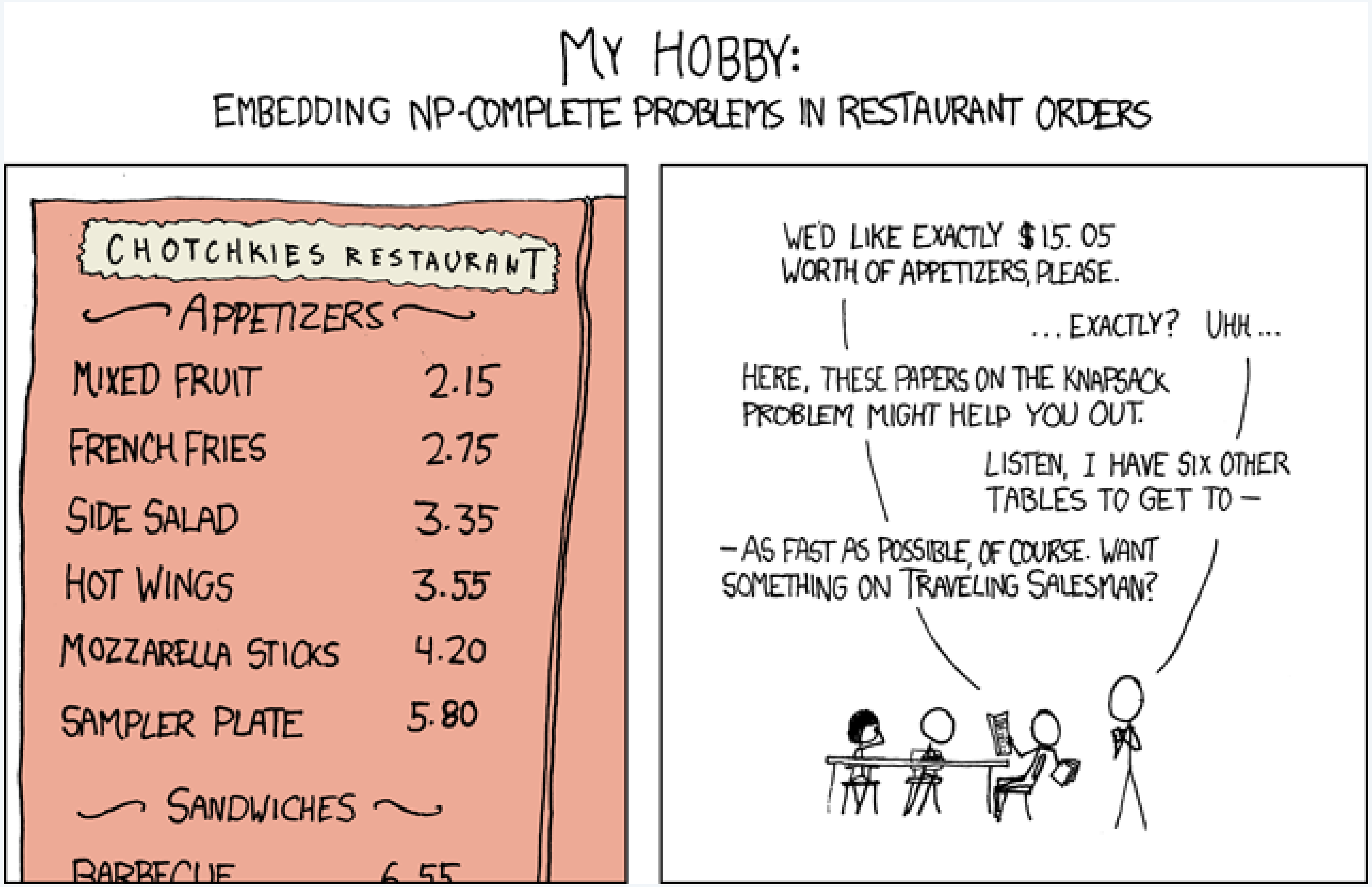
3-SAT instance

dummies to get clause columns to sum to 4

	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
	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

SUBSET-SUM instance

My hobby



Randall Munro

http://xkcd.com/c287.html

Partition

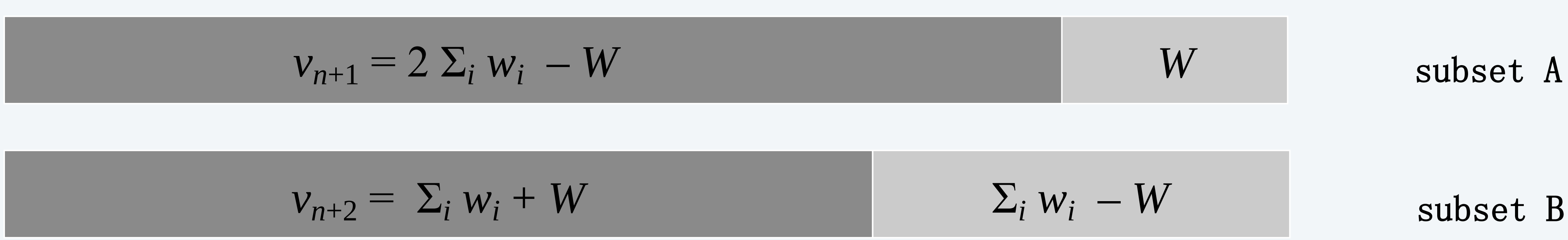
SUBSET-SUM. Given natural numbers $w_1, \ldots, w_n$ and an integer W, is there a subset that adds up to exactly W?

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

Theorem. SUBSET-SUM $\leq_P$ PARTITION.

Pf. Let $W, w_1, \ldots, w_n$ be an instance of SUBSET-SUM.

- Create instance of PARTITION with $m = n + 2$ elements.
  - $v_1 = w_1, v_2 = w_2, \ldots, v_n = w_n, \quad v_{n+1} = 2 \Sigma_i w_i - W, \quad v_{n+2} = \Sigma_i w_i + W$
- 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. ▪



SCHEDULE. Given a set of n jobs with processing time t_j, release time r_j, and deadline d_j, is it possible to schedule all jobs on a single machine such that job j is processed with a contiguous slot of t_j time units in the interval $[r_j, d_j]$?

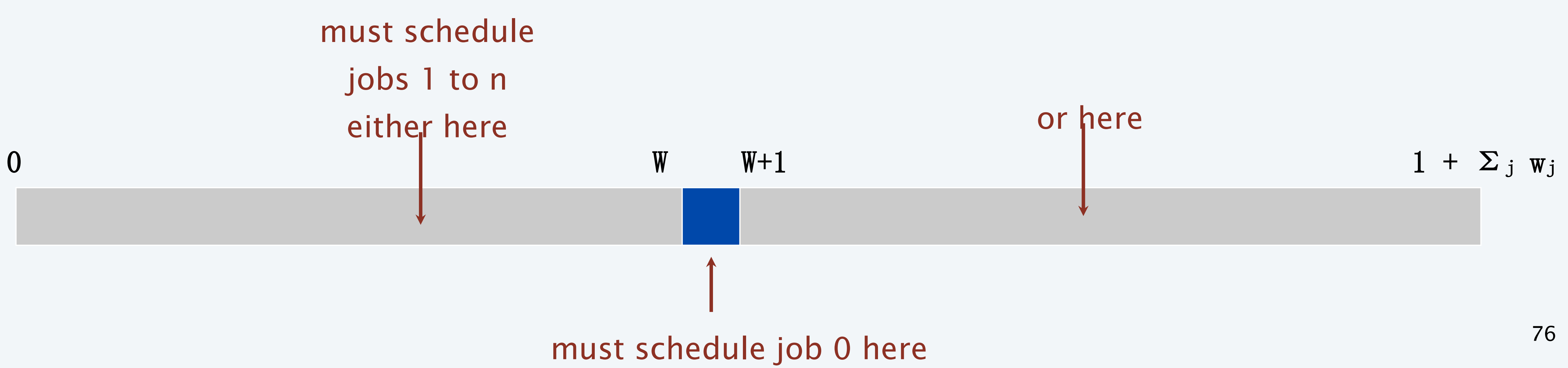
Ex.

Theorem. SUBSET-SUM $\leq_P$ SCHEDULE.

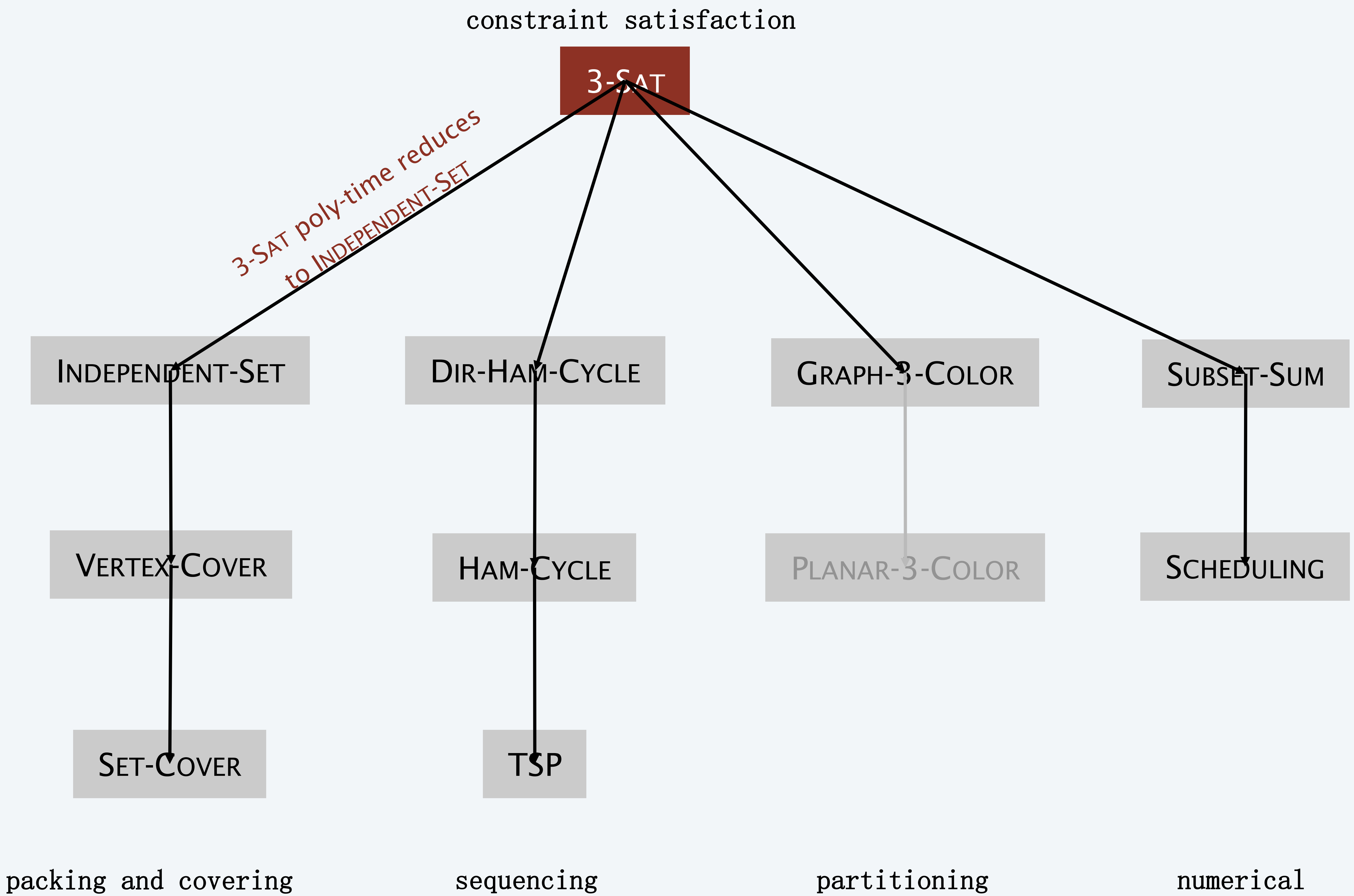
Pf. Given SUBSET-SUM instance $w_1, \ldots, w_n$ and target W, construct an instance of SCHEDULE that is feasible iff there exists a subset that sums to exactly W.

Construction.

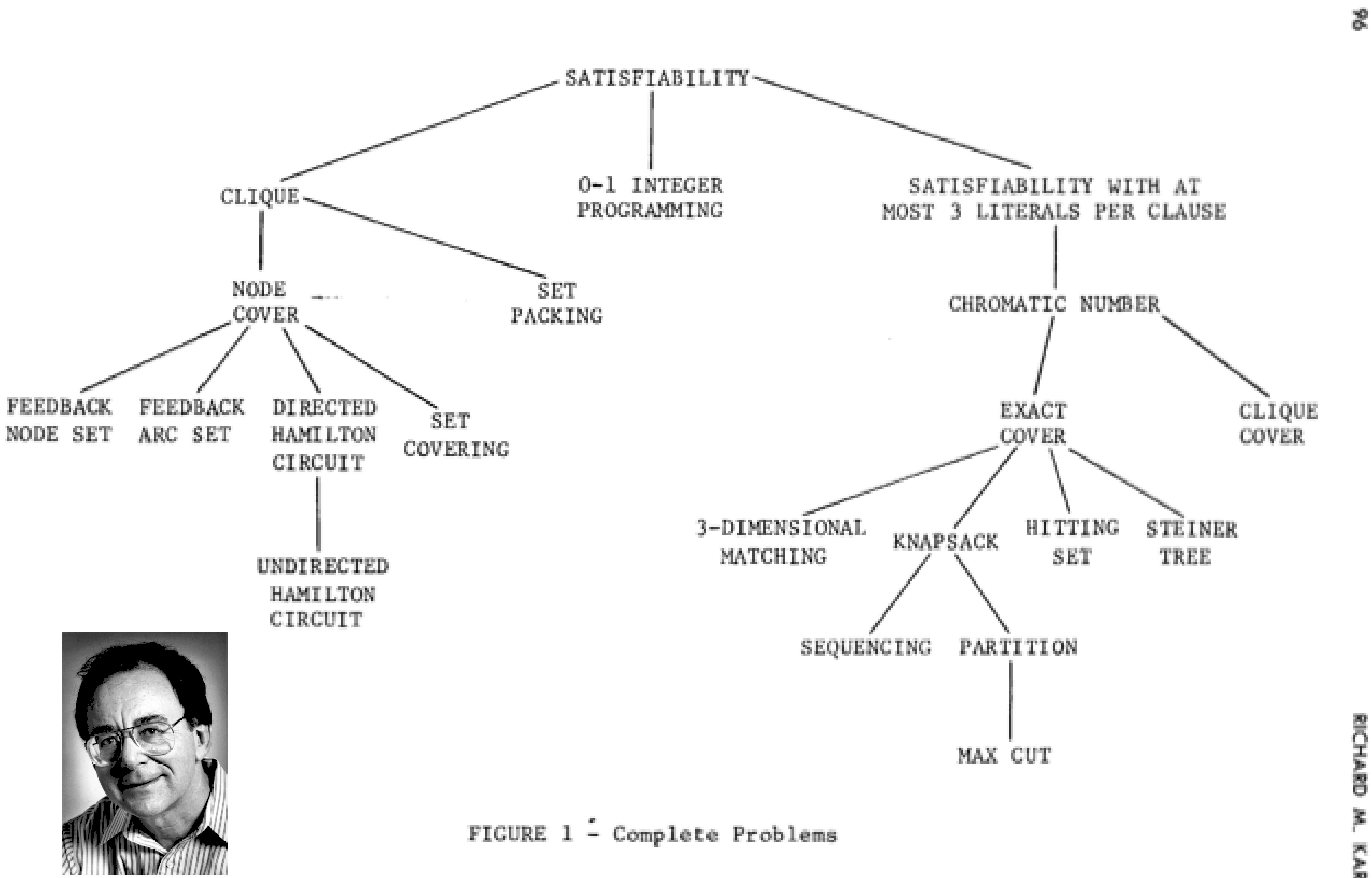
- Create n jobs with processing time $t_j = w_j$, release time $r_j = 0$, and no deadline ($d_j = 1 + \Sigma_j w_j$).
- Create job 0 with $t_0 = 1$, release time $r_0 = W$, and deadline $d_0 = W + 1$.
- Lemma: subset that sums to W iff there exists a feasible schedule. ▪



Polynomial-time reductions



Karp's 21 NP-complete problems



Dick Karp (1972)

1985 Turing Award