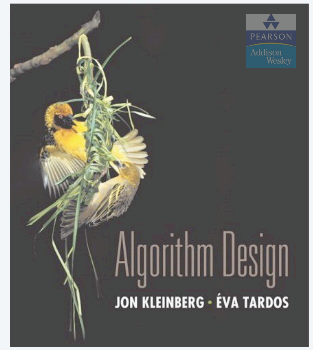
Intractability

Problem Reductions

Tyler Moore

CS 2123, The University of Tulsa

Some slides created by or adapted from Dr. Kevin Wayne. For more information see `http://www.cs.princeton.edu/~wayne/kleinberg-tardos`. Some code reused from Python Algorithms by Magnus Lie Hetland.



SECTION 8.1

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.

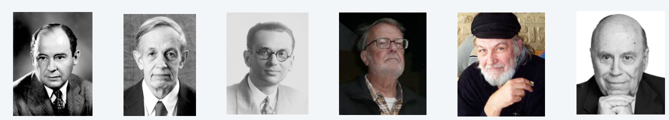
Algorithm design antipatterns.

- 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.

Practice. Poly-time algorithms scale to huge problems.

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

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.

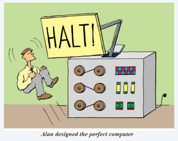
yes	probably no
shortest path	longest path
min cut	max cut
2-satisfiability	3-satisfiability
planar 4-colorability	planar 3-colorability
bipartite vertex cover	vertex cover
matching	3d-matching
primality testing	factoring
linear programming	integer linear programming

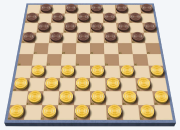
Classify problems

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

Provably requires exponential time.

- Given a constant-size program, does it halt in at most k steps? (input size = c + lg k)
- Given a board position in an n-by-n generalization of checkers, can black guarantee a win? (using forced capture rule)





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

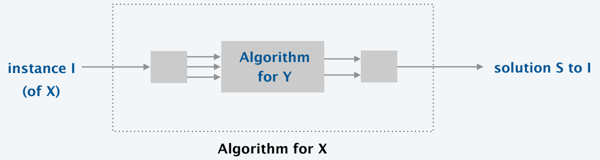
Polynomial-time reductions

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



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.

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 $Y \leq_P X$.

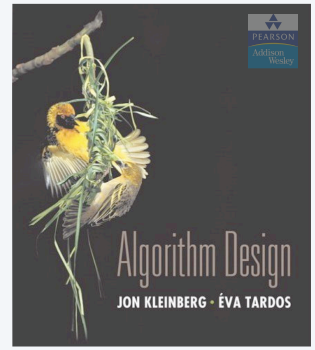
Polynomial-time reductions

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

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

Establish equivalence. If both $X \le_P Y$ and $Y \le_P X$, 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.



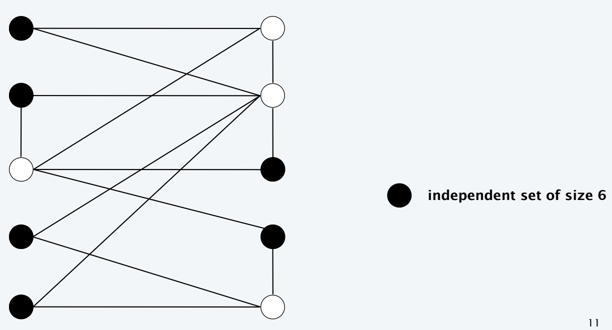
SECTION 8.1

8. INTRACTABILITY I

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 $|S| \ge k$, and for each edge at most one of its endpoints is in S?

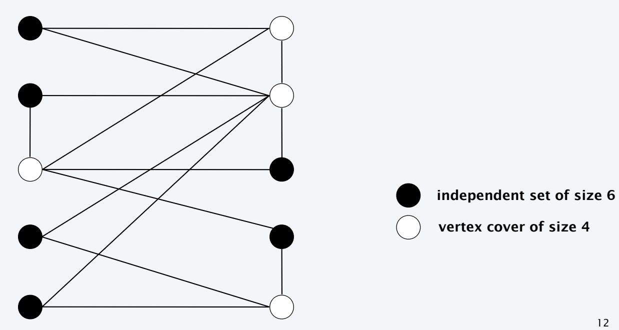
Ex. Is there an independent set of size ≥ 6?
Ex. Is there an independent set of size ≥ 7?



Vertex cover

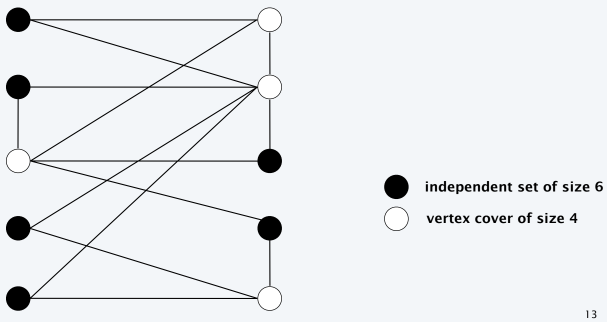
VERTEX-COVER. Given a graph $G = (V, E)$ and an integer k, is there a subset of vertices $S \subseteq V$ such that $|S| \le k$, and for each edge, at least one of its endpoints is in S?

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 $n - k$.

$\Rightarrow$

- 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. 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 $n - k$.

$\Leftarrow$

- 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. ▪

Set cover

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.

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

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

$k = 2$

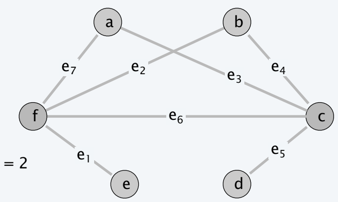
a set cover instance

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 $v \in V$: $S_v = \{e \in E : e \text{ incident to } v\}$.

k = 2



vertex cover instance
(k = 2)

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

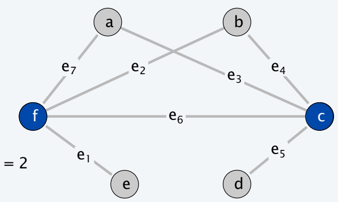
set cover instance
(k = 2)

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. ▪

k = 2



vertex cover instance
(k = 2)

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

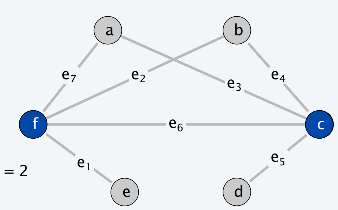
set cover instance
(k = 2)

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. ▪

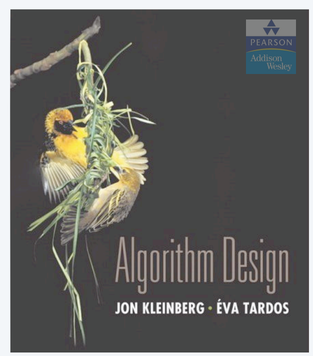
k = 2



vertex cover instance
(k = 2)

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

set cover instance
(k = 2)



SECTION 8.2

8. INTRACTABILITY I

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

Satisfiability

Literal. A boolean variable or its negation. x_i 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

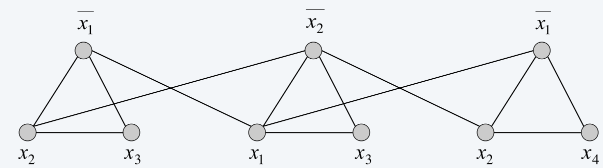
Key application. Electronic design automation (EDA).

3-satisfiability reduces to independent set

Theorem. 3-SAT $\le_P$ 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

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

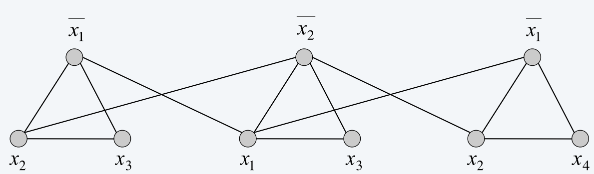
3-satisfiability reduces to independent set

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

G



k = 3

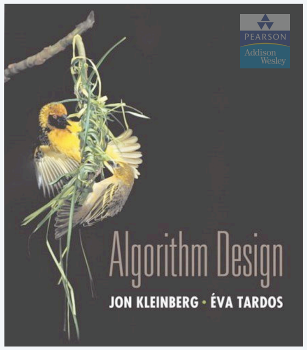
$$\Phi = (\overline{x_1} \vee x_2 \vee x_3) \wedge (x_1 \vee \overline{x_2} \vee x_3) \wedge (\overline{x_1} \vee x_2 \vee x_4)$$

Review

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

Transitivity. If $X \le_P Y$ and $Y \le_P Z$, then $X \le_P Z$.
Pf idea. Compose the two algorithms.

Ex. 3-SAT $\le_P$ INDEPENDENT-SET $\le_P$ VERTEX-COVER $\le_P$ SET-COVER.



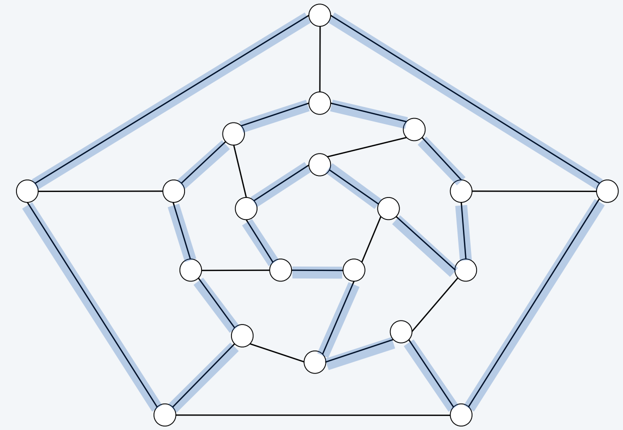
SECTION 8.5

8. INTRACTABILITY I

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

Hamilton cycle

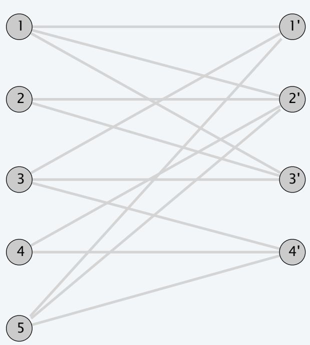
HAM-CYCLE. Given an undirected graph $G = (V, E)$, does there exist a simple cycle Γ that contains every node in V?



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?



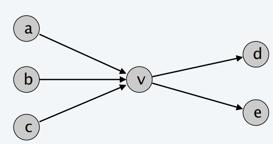
no

Directed hamilton cycle reduces to hamilton cycle

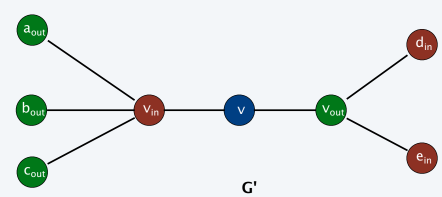
DIR-HAM-CYCLE: Given a digraph $G = (V, E)$, does there exist a simple directed cycle Γ that contains every node in V?

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

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



G



G'

Directed hamilton cycle reduces to hamilton cycle

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

Pf. $\Rightarrow$

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

Pf. $\Leftarrow$

- 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 $\leq_P$ DIR-HAM-CYCLE.

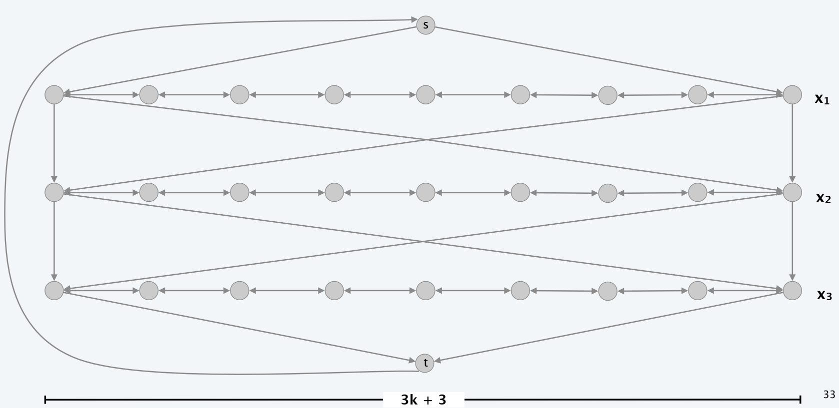
Pf. Given an instance Φ of 3-SAT, we construct an instance of DIR-HAM-CYCLE that has a Hamilton cycle iff Φ is satisfiable.

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.

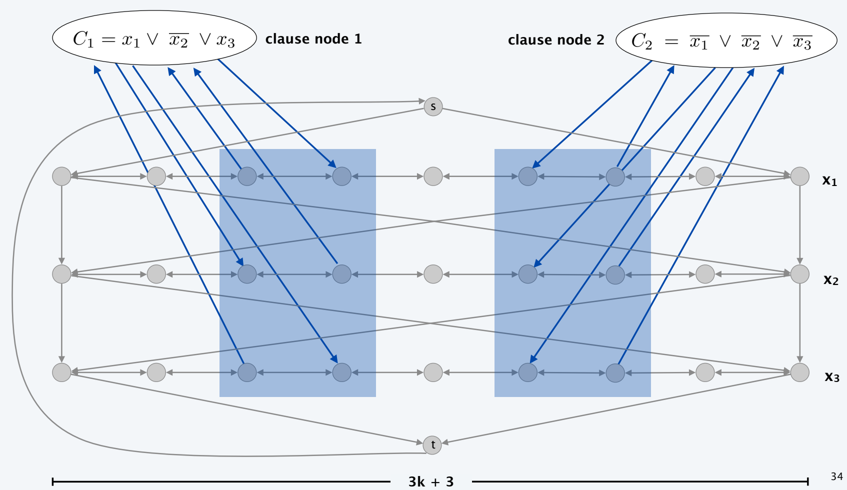
- Construct G to have 2^n Hamilton cycles.
- Intuition: traverse path i from left to right $\Leftrightarrow$ set variable $x_i = true$.



3-satisfiability reduces to directed hamilton 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-satisfiability reduces to directed hamilton cycle

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

Pf. $\Rightarrow$

- 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. $\Leftarrow$

- 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

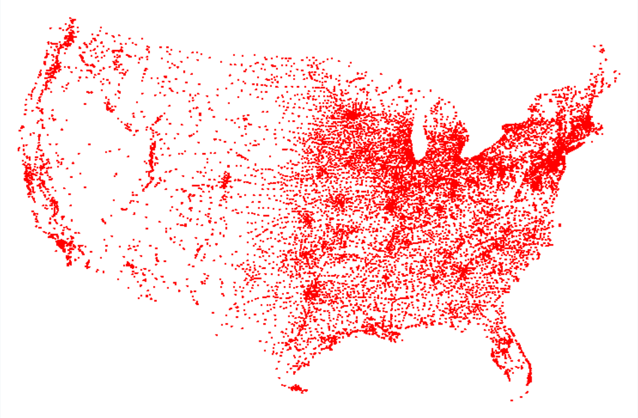
LONGEST-PATH. Given a directed graph $G = (V, E)$, does there exists a simple path consisting of at least k edges?

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

Pf 1. Redo proof for DIR-HAM-CYCLE, ignoring back-edge from t to s.
Pf 2. Show HAM-CYCLE $\le_P$ LONGEST-PATH.

Traveling salesperson problem

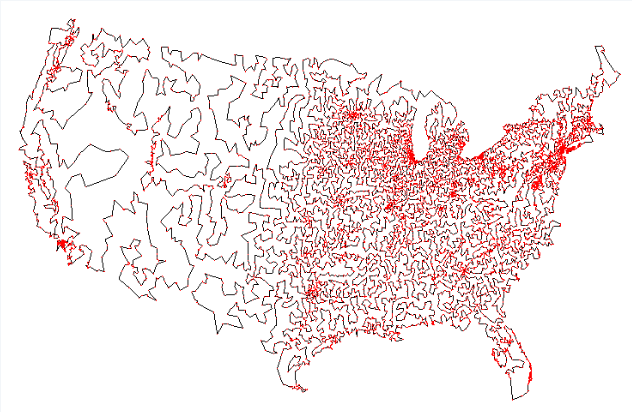
TSP. Given a set of n cities and a pairwise distance function $d(u, v)$, is there a tour of length $\le D$?



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

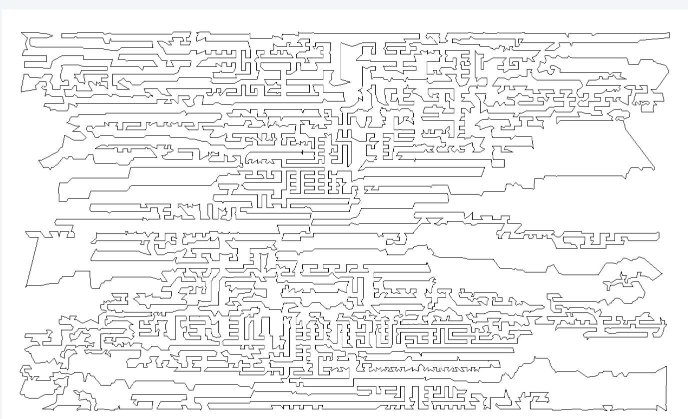
Traveling salesperson problem

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.

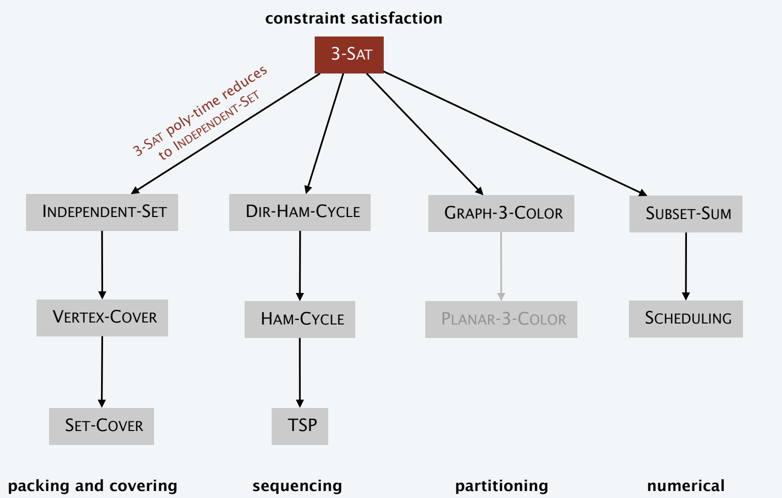
- Given instance $G = (V, E)$ of HAM-CYCLE, create n cities with distance function

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

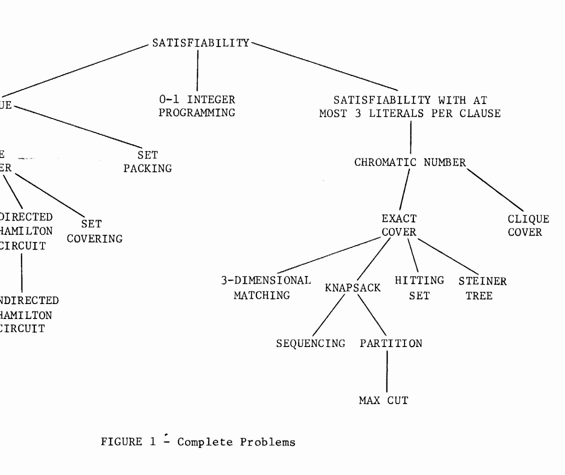
- TSP instance has tour of length $\leq n$ iff G has a Hamilton cycle. ▪

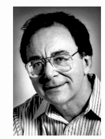
Remark. TSP instance satisfies triangle inequality: $d(u, w) \leq d(u, v) + d(v, w)$.

Polynomial-time reductions



Karp's 21 NP-complete problems





Dick Karp (1972)
1985 Turing Award