NP completeness and computational tractability Part II

Grand challenge: Classify Problems According to Computational Requirements

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

A working definition. [Cobham 1964, Edmonds 1965, Rabin 1966]
Those with polynomial-time algorithms.

Yes	Probably no
Shortest path	Longest path
Matching	3D-matching
Min cut	Max cut
2-SAT	3-SAT
Planar 4-color	Planar 3-color
Bipartite vertex cover	Vertex cover
Primality testing	Factoring

Classify Problems

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

For any nice function T(n)
There are problems that require more than T(n) time to solve.

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

NP-completeness: Show that these fundamental problems are "computationally equivalent" and appear to be different manifestations of one really hard problem.

Polynomial-Time Reduction

Desiderata'. Suppose we could solve X in polynomial-time. What else could we solve in polynomial time?

Reduction. Problem X polynomial 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. $X \leq_P Y$.

Polynomial-Time Reduction

Purpose. Classify problems according to relative difficulty.

Design algorithms. If $X \leq_P Y$ and Y can be solved in polynomial-time, then X can also 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 $X \leq_P Y$ and $Y \leq_P X$, we use notation $X \equiv_P Y$.
↑
up to cost of reduction

Basic Reduction Strategies

- Reduction by simple equivalence.
- Reduction from special case to general case.
- Reduction by encoding with gadgets.

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.

Self-Reducibility

Decision problem. Does there exist a vertex cover of size $\le k$?
Search problem. Find vertex cover of minimum cardinality.

Self-reducibility. Search problem $\le_P$ decision version.

- Applies to all (NP-complete) problems we discuss.
- Justifies our focus on decision problems.

Ex: to find min cardinality vertex cover.

- (Binary) search for cardinality k^* of min vertex cover.
- Find a vertex v such that $G - \{ v \}$ has a vertex cover of size $\le k^* - 1$.
  - any vertex in any min vertex cover will have this property
- Include v in the vertex cover.
- Recursively find a min vertex cover in $G - \{ v \}$.

Definition of NP

Decision Problems

Decision problem.

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

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

length of s

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

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 $C(s, t) = \text{yes}$.

"certificate" or "witness"

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

$C(s, t)$ is a poly-time algorithm and $|t| \le p(|s|)$ for some polynomial $p(\cdot)$.

Remark. NP stands for nondeterministic polynomial-time.

NP -- another definition

Nondeterministic Turing machines

- At any point in a computation, the machine may proceed according to several possibilities.

Machine accepts if there is a computation branch that ends in an accepting state.

Example: NTM for Clique
On input (G,k) where G is a graph
- Nondeterministically select a subset S of k nodes of G
- Test whether G contains all edges connecting nodes in S.
- If yes, *accept*, else *reject*.

Theorem: A language is in NP iff it is decided by some nondeterministic polynomial time Turing machine.

NP -- equivalence of definitions

Theorem: A language is in NP iff it is decided by some nondeterministic polynomial time Turing machine

Proof: let A be a language in NP.

=> Let C(s,t) be a certifer for A that runs in time n^k.
Construct nondeterministic TM N that on input s of length n does:

- Nondeterministically select string t of length at most n^k
- Run C(s,t)
- If C(s,t) accepts, *accept*, otherwise *reject*

<= Suppose N is a NTM that decides A. Construct verifier C that on input (s,t) does the following:

- Simulate N on input s, treating each symbol of t as a description of the nondeterministic choice to make at each step.
- If this branch of N's computation accepts, *accept*, else *reject*.

NP

Theorem: A language is in NP iff it is decided by some nondeterministic polynomial time Turing machine

NTIME (t(n)) = {L | L is a language decided by an O(t(n)) time nondeterministic Turing machine}

NP = $\cup_k$ NTIME (n^k) = languages with poly-time verifiers

P, NP, EXP

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

Claim. P $\subseteq$ NP.
Pf. Consider any problem X in P.

- By definition, there exists a poly-time algorithm A(s) that solves X.
- Certificate: t = ε, certifier C(s, t) = A(s). ▪

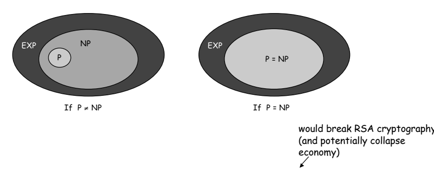
Claim. NP $\subseteq$ EXP.
Pf. Consider any problem X in NP.

- By definition, there exists a poly-time certifier C(s, t) for X.
- To solve input s, run C(s, t) on all strings t with $|t| \le p(|s|)$.
- 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 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.

NP-Complete

NP-complete. A problem Y in NP with the property that for every problem X in NP, $X \le_p Y$.

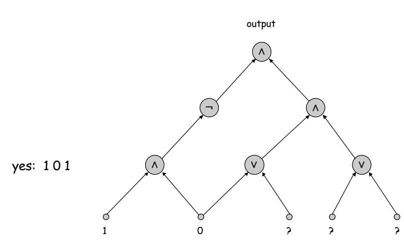
Theorem. Suppose Y is an NP-complete problem. Then Y is solvable in poly-time iff P = NP.
Pf. $\Leftarrow$ If P = NP then Y can be solved in poly-time since Y is in NP.
Pf. $\Rightarrow$ Suppose Y can be solved in poly-time.

- Let X be any problem in NP. Since $X \le_p Y$, 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?



The "First" NP-Complete Problem

Theorem. CIRCUIT-SAT is NP-complete. [Cook 1971, Levin 1973]
Pf. (sketch++)

- 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, i.e. Turing machine on |s| + p(|s|) bits (input s, certificate t)
- Assumptions about TM:
  - It moves its head all the way to left and writes blank in leftmost tape cell right before halting.
  - Once it halts, it stays in same configuration for all future steps.
- Convert TM it into a poly-size circuit K.
  - first |s| bits are hard-coded with s
  - remaining p(|s|) bits represent bits of t
- Construct circuit K that is satisfiable iff C(s, t) = yes.

19

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 $\le_P$ Y.

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

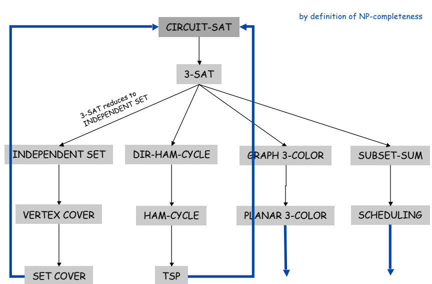
Pf. Let W be any problem in NP. Then W $\underbrace{\le_P}_{\text{by definition of NP-complete}}$ X $\underbrace{\le_P}_{\text{by assumption}}$ Y.

- By transitivity, W $\le_P$ Y.
- Hence Y is NP-complete. ▪

20

NP-Completeness

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



21

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.

22

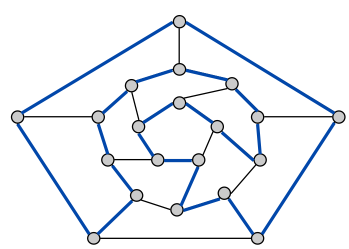
Sequencing Problems

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.

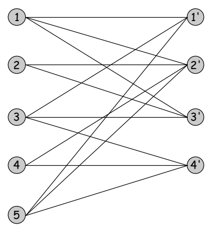


YES: vertices and faces of a dodecahedron.

24

Hamiltonian Cycle

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



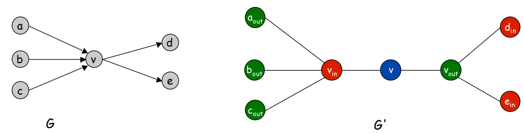
NO: bipartite graph with odd number of nodes.

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.



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

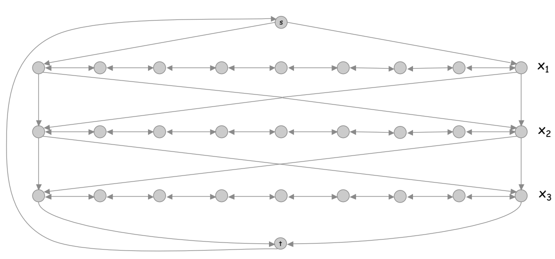
Claim. 3-SAT $\leq_P$ DIR-HAM-CYCLE.

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.

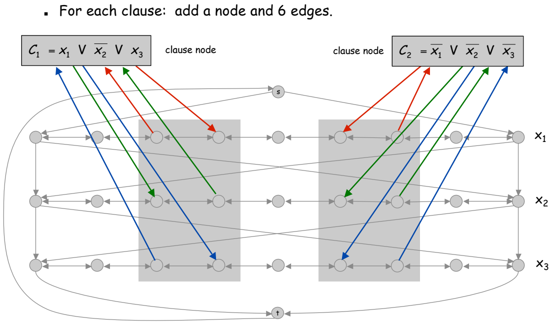
3-SAT Reduces to Directed Hamiltonian Cycle

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?

Prove that LONGEST-PATH is NP-complete

The Longest Path †

Lyrics. Copyright © 1988 by Daniel J. Barrett.
Music. Sung to the tune of *The Longest Time* by Billy Joel.
http://www.cs.princeton.edu/~wayne/cs423/lectures/longest-path.mp3

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

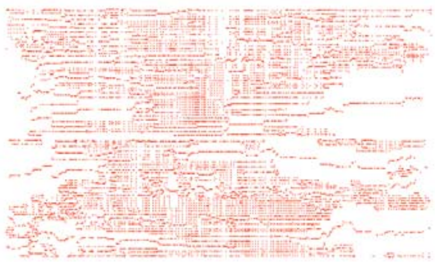
Traveling Salesperson Problem

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

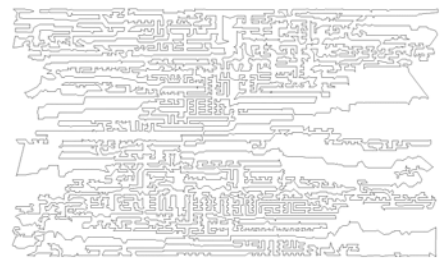
TSP. Given a set of n cities and a pairwise distance function d(u, v), is there a tour of length ≤ D?



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 ≤ 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 ≤ D?

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

Claim. HAM-CYCLE $\le_P$ 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 ≤ n iff G is Hamiltonian. ▪

Remark. TSP instance in reduction satisfies Δ-inequality.

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?

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 126	TTh 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

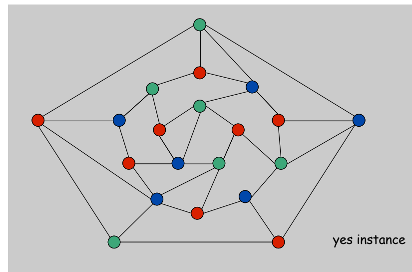
3D-MATCHING. Given disjoint sets X, Y, and Z, each of size n and a set $T \subseteq X \times Y \times Z$ of triples, does there exist a set of n triples in T such that each element of $X \cup Y \cup Z$ is in exactly one of these triples?

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

3-Colorability

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

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

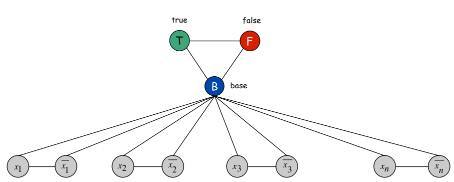
Construction.
- i. For each literal, create a node.
- ii. Create 3 new nodes T, F, B; connect them in a triangle, and connect each literal to B.
- iii. Connect each literal to its negation.
- iv. For each clause, add gadget of 6 nodes and 13 edges.

↑ to be described next

3-Colorability

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

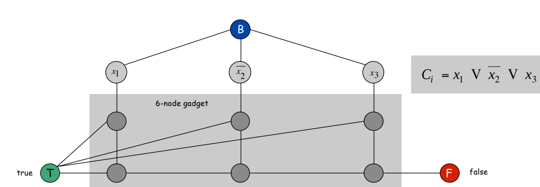
Pf. $\Rightarrow$ Suppose graph is 3-colorable.
- Consider assignment that sets all T literals to true.
- (ii) ensures each literal is T or F.
- (iii) ensures a literal and its negation are opposites.



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.

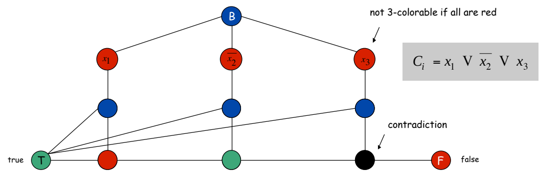


3-Colorability

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.

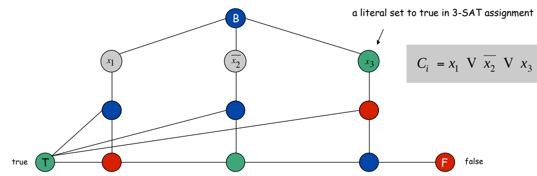


3-Colorability

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. ▪



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

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.

$$C_1 = \bar{x} \vee y \vee z$$
$$C_2 = x \vee \bar{y} \vee z$$
$$C_3 = \bar{x} \vee \bar{y} \vee \bar{z}$$

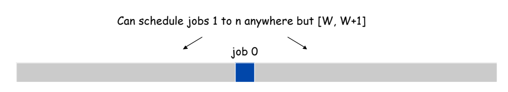
	x	y	z	C_1	C_2	C_3	
x	1	0	0	0	1	0	100,010
¬ x	1	0	0	1	0	1	100,101
y	0	1	0	1	0	0	10,100
¬ y	0	1	0	0	1	1	10,011
z	0	0	1	1	1	0	1,110
¬ z	0	0	1	0	0	1	1,001
dummies to get clause columns to sum to 4	0	0	0	1	0	0	100
	0	0	0	2	0	0	200
	0	0	0	0	1	0	10
	0	0	0	0	2	0	20
	0	0	0	0	0	1	1
	0	0	0	0	0	2	2
W	1	1	1	4	4	4	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 $\le_P$ SCHEDULE-RELEASE-TIMES.
Pf. Given an instance of SUBSET-SUM $w_1, \ldots, w_n$, and target W,

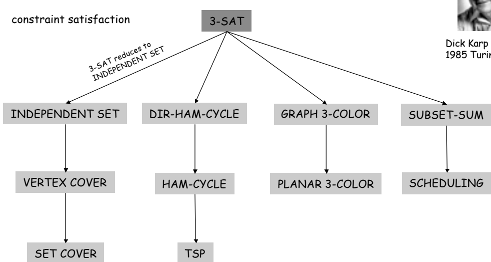
- Create n jobs with processing time $t_i = w_i$, release time $r_i = 0$, and no deadline ($d_i = 1 + \Sigma_j w_j$).
- Create job 0 with $t_0 = 1$, release time $r_0 = W$, and deadline $d_0 = W+1$.



Polynomial-Time Reductions

Dick Karp (1972)
1985 Turing Award

constraint satisfaction



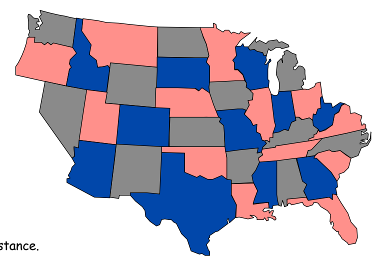
packing and covering | sequencing | partitioning | numerical

55

An Extra : 4 Color Theorem

Planar 3-Colorability

PLANAR-3-COLOR. Given a planar map, can it be colored using 3 colors so that no adjacent regions have the same color?

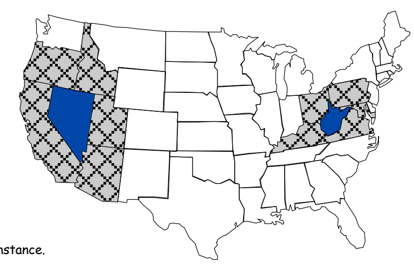


YES instance.

57

Planar 3-Colorability

PLANAR-3-COLOR. Given a planar map, can it be colored using 3 colors so that no adjacent regions have the same color?



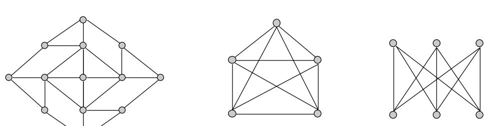
NO instance.

58

Planarity

Def. A graph is planar if it can be embedded in the plane in such a way that no two edges cross.

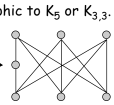
Applications: VLSI circuit design, computer graphics.



Planar | K_5: non-planar | $K_{3,3}$: non-planar

Kuratowski's Theorem. An undirected graph G is non-planar iff it contains a subgraph homeomorphic to K_5 or $K_{3,3}$.

homeomorphic to $K_{3,3}$ →



59

Planarity Testing

Planarity testing. [Hopcroft-Tarjan 1974] $O(n)$.

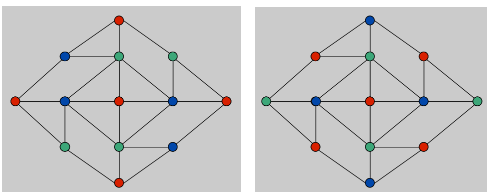
↑ simple planar graph can have at most $3n$ edges

Remark. Many intractable graph problems can be solved in poly-time if the graph is planar; many tractable graph problems can be solved faster if the graph is planar.

60

Planar 3-Colorability

Claim. 3-COLOR $\le_p$ PLANAR-3-COLOR.
Proof sketch: Given instance of 3-COLOR, draw graph in plane, letting edges cross if necessary.
- Replace each edge crossing with the following planar gadget W.
  - in any 3-coloring of W, opposite corners have the same color
  - any assignment of colors to the corners in which opposite corners have the same color extends to a 3-coloring of W

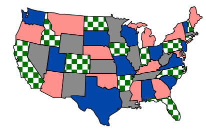


Planar k-Colorability

PLANAR-2-COLOR. Solvable in linear time.

PLANAR-3-COLOR. NP-complete.

PLANAR-4-COLOR. Solvable in O(1) time.



Theorem. [Appel-Haken, 1976] Every planar map is 4-colorable.
- Resolved century-old open problem.
- Used 50 days of computer time to deal with many special cases.
- First major theorem to be proved using computer.

False intuition. If PLANAR-3-COLOR is hard, then so is PLANAR-4-COLOR and PLANAR-5-COLOR.

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. NON-SATISFIABLE.
- 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 $\equiv_p$ NON-SATISFIABLE, but how do we classify NON-SATISFIABLE?

↑
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 reverse.

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. NON-SATISFIABLE, NO-HAM-CYCLE, PRIMES.

Why doesn't the following poly-time NTM solve NON-SATISFIABILITY?
- Guess an assignment to variables.
- If assignment doesn't satisfy formula, *accept*. Else, *reject*

NP and co-NP

NP. Decision problems for which there is a poly-time certifier.
Def. Given a decision problem X, its complement $\overline{X}$ is the same problem with the yes and no answers reverse.
co-NP. Complements of decision problems in NP.

An equivalent definition:

co-NP:
We say that L is in co-NP if there is a polynomial p, and a poly time TM M, such that for every x,

x is in L iff **for all** u of length p(|x|) M(x,u)=1

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 ⊆ NP ∩ co-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. Does P = NP ∩ co-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 ∩ co-NP, but not known to be in P.

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

PRIMES is in NP ∩ co-NP

Theorem. PRIMES is in NP ∩ co-NP.
Pf. We already know that PRIMES is in co-NP, so it suffices to prove that PRIMES is in NP.

Pratt's Theorem. An odd integer s is prime iff there exists an integer 1 < t < s s.t.

$$t^{s-1} \equiv 1 \pmod{s}$$
$$t^{(s-1)/p} \not\equiv 1 \pmod{s}$$
for all prime divisors p of s-1

Input. s = 437,677
Certificate. t = 17, $2^2 \times 3 \times 36{,}473$

prime factorization of s-1
also need a recursive certificate
to assert that 3 and 36,473 are prime

Certifier.
- Check s-1 = 2 × 2 × 3 × 36,473.
- Check $17^{s-1} = 1 \pmod{s}$.
- Check $17^{(s-1)/2} \equiv 437{,}676 \pmod{s}$.
- Check $17^{(s-1)/3} \equiv 329{,}415 \pmod{s}$.
- Check $17^{(s-1)/36{,}473} \equiv 305{,}452 \pmod{s}$.

use repeated squaring

FACTOR is in NP ∩ co-NP

FACTORIZE. Given an integer x, find its prime factorization.
FACTOR. Given two integers x and y, does x have a nontrivial factor less than y?

Theorem. FACTOR $\equiv_P$ FACTORIZE.

Theorem. FACTOR is in NP ∩ co-NP.
Pf.

- Certificate: a factor p of x that is less than y.
- Disqualifier: the prime factorization of x (where each prime factor is less than y), along with a certificate that each factor is prime.

Primality Testing and Factoring

We established: PRIMES $\le_P$ COMPOSITES $\le_P$ FACTOR.

Natural question: Does FACTOR $\le_P$ PRIMES ?
Consensus opinion. No.

State-of-the-art.

- PRIMES is in P. ← proved in 2001
- FACTOR not believed to be in P.

RSA cryptosystem.

- Based on dichotomy between complexity of two problems.
- To use RSA, must generate large primes efficiently.
- To break RSA, suffices to find efficient factoring algorithm.

Some Philosophical Remarks

P v NP captures important philosophical phenomenon: recognizing the correctness of an answer is often easier than coming up with the answer.

P v NP asks if exhaustive search can be avoided.

If P = NP -- then there is an algorithm that finds mathematical proofs in time polynomial in the length of the proof.

Theorems = $\{(\varphi, 1^n) : \varphi$ has a formal proof of length at most n in axiomatic system A$\}$

Theorems is in NP

In fact, Theorems is NP-complete.

73

The Riemann Hypothesis

Considered by many mathematicians to be the most important unresolved problem in pure mathematics

Conjecture about the distribution of zeros of the Riemann zeta - function

1 Million dollar prize offered by Clay Institute

74

3D Bin Packing is NP-Complete

There is a finite and not unimaginably large set of boxes, such that if we knew how to pack those boxes into the trunk of your car, then we'd also know a proof of the Riemann Hypothesis. Indeed, *every* formal proof of the Riemann Hypothesis with at most (say) a million symbols corresponds to some way of packing the boxes into your trunk, and vice versa. Furthermore, a list of the boxes and their dimensions can be feasibly written down.

Courtesy of Scott Aaronson

75

Why do we believe P different from NP if we can't prove it?

- The empirical argument: hardness of solving NP-complete problems in practice.
- There are "vastly easier" problems than NP-complete ones (like factoring) that we already have no idea how to solve in P
- P=NP would mean that mathematical creativity could be automated. "God would not be so kind!" Scott Aaronson

- We will add to this list later…

76

Why is it so hard to prove P different from NP?

- Because P is different from NP
- Because there are lots of clever, non-obvious polynomial time algorithms. For example, proof that 3SAT is hard will have to fail for 2-SAT. Proof that 3-coloring planar graphs is hard will have to fail for 4-coloring planar graphs. Etc Etc.

- We'll add to this list later….

77