CSCI 355: Algorithm Design and Analysis
10. Intractability

- *poly-time reductions*
- *packing and covering problems*
- *constraint satisfaction problems*
- *sequencing problems*
- *graph colouring*
- *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.

Classifying problems according to computational requirements

Q. Which problems will we be able to solve in practice?
A working definition. Those with poly-time algorithms.

Turing machine, word RAM, uniform circuits, …

Theory. Definition is broad and robust.

constants tend to be small, e.g., $3n^2$

Practice. Poly-time algorithms scale to huge problems.

Classifying problems according to computational requirements

Q. Which problems will we be able to solve in practice?
A working definition. Those with poly-time algorithms.

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

Classifying problems

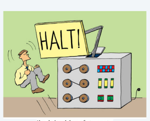
Desiderata. Classify problems according to those that can be solved in polynomial time and those that cannot.

Problems that provably require exponential time.

input size = $c + \log k$

- 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



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

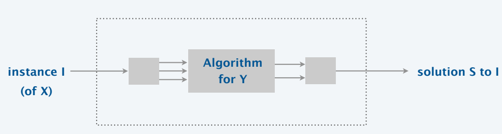
Poly-time reductions

Precise desiderata. Suppose we could solve a problem Y in polynomial time. What other problems could we solve in polynomial time?

Reduction. Problem X is polynomial-time reducible to problem Y if arbitrary instances of problem X can be solved using:

- a polynomial number of standard computational steps, plus
- a polynomial number of calls to an oracle that solves problem Y.

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



Poly-time reductions

Precise desiderata. Suppose we could solve a problem Y in polynomial time. What other problems could we solve in polynomial time?

Reduction. Problem X is polynomial-time reducible to problem Y if arbitrary instances of problem X can be solved using:

- a polynomial number of standard computational steps, plus
- a polynomial number of calls to an oracle that solves problem Y.

Notation. $X \leq_P Y$.

Note. We pay for the time to write down instances of Y sent to oracle $\Rightarrow$ instances of Y must be of polynomial size.

Common mistake. Confusing $X \leq_P Y$ with $Y \leq_P X$.

Poly-time reductions

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

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

Proving equivalence. If both $X \leq_P Y$ and $Y \leq_P X$, then X can be solved in polynomial time iff Y can be solved in polynomial time; we write $X \equiv_P Y$.

Bottom line. Reductions classify problems according to relative difficulty.

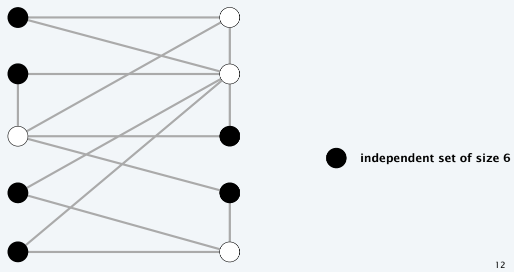
CSCI 355: Algorithm Design and Analysis
10. Intractability

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

Independent set

INDEPENDENT-SET. Given a graph $G = (V, E)$ and an integer k, is there a subset of k (or more) vertices such that no two are adjacent?

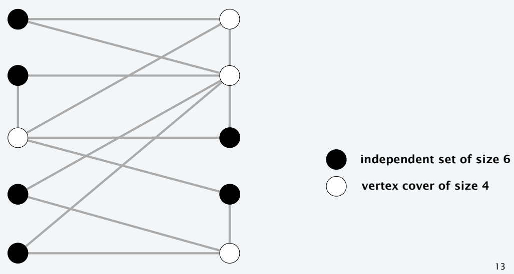
Ex. Is there an independent set of size ≥ 6 ?
Ex. Is there an independent set of size ≥ 7 ?



Vertex cover

VERTEX-COVER. Given a graph $G = (V, E)$ and an integer k, is there a subset of k (or fewer) vertices such that each edge is incident to at least one vertex in the subset?

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

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 $(u, v) \in E$.
- S is independent ⇒ either $u \notin S$, or $v \notin S$, or both.
  ⇒ 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. INDEPENDENT-SET $\equiv_P$ VERTEX-COVER.

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

[⇐]:

- Let $V - S$ be any vertex cover of size $n - k$.
- S is of size k.
- Consider an arbitrary edge $(u, v) \in E$.
- $V - S$ is a vertex cover ⇒ either $u \in V - S$, or $v \in V - S$, or both.
  ⇒ either $u \notin S$, or $v \notin S$, or both.
- Thus, S is an independent set. ▪

Set cover

SET-COVER. Given a set U of elements, a collection S of subsets of U, and an integer k, are there $\le k$ of these subsets whose union is equal to U?

Ex.

- 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.

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

a set cover instance

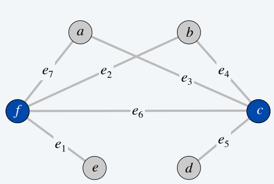
Vertex cover reduces to set cover

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

Pf. Given a VERTEX-COVER instance $G = (V, E)$ and an integer k, we construct a SET-COVER instance (U, S, k) that has a set cover of size k iff G has a vertex cover of size k.

Construction.

- Take the universe $U = E$.
- Include one subset for each vertex $v \in V$: $S_v = \{e \in E : e \text{ incident to } v \}$.



vertex cover instance
(k = 2)

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

set cover instance
(k = 2)

CSCI 355: Algorithm Design and Analysis
10. Intractability

- poly-time reductions
- packing and covering problems
- constraint satisfaction problems
- sequencing problems
- graph colouring
- 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 (CNF). A propositional formula Φ that is a conjunction of clauses. $\Phi = C_1 \wedge C_2 \wedge C_3 \wedge C_4$

SAT. Given a CNF formula Φ, does it have a satisfying truth assignment?

3-SAT. An instance of SAT where each clause contains exactly 3 literals (and each literal corresponds to a different variable).

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

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

Satisfiability is hard

Scientific hypothesis. There does not exist a poly-time algorithm for 3-SAT.

P vs. NP. This hypothesis is equivalent to the **P ≠ NP** conjecture.

"I can't find an efficient algorithm, but neither can all these famous people."

3-satisfiability reduces to independent set

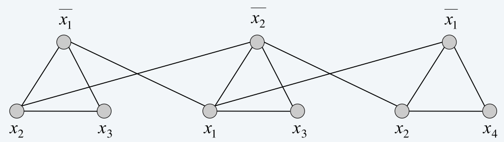
Theorem. 3-SAT $\leq_P$ INDEPENDENT-SET.

Pf. Given an instance Φ of 3-SAT, we construct an instance (G, k) of INDEPENDENT-SET that has an independent set of size $k = |\Phi|$ iff Φ is satisfiable.

Construction.

- G contains 3 vertices for each clause, one for each literal.
- Connect 3 literals in a clause in a triangle.
- Connect each literal to each of its negations.

G



k = 3

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

Review

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 sketch. Compose the two algorithms.

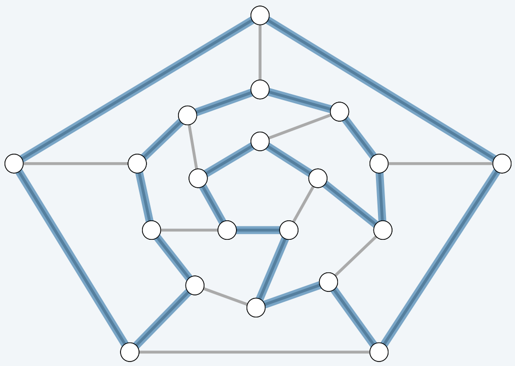
Ex. 3-SAT $\leq_P$ INDEPENDENT-SET $\leq_P$ VERTEX-COVER $\leq_P$ SET-COVER.

CSCI 355: ALGORITHM DESIGN AND ANALYSIS
10. INTRACTABILITY

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

Hamiltonian cycle

HAMILTON-CYCLE. Given an undirected graph $G = (V, E)$, does there exist a cycle Γ that visits every vertex exactly once?



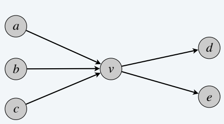
yes

Directed Hamiltonian cycle reduces to Hamiltonian cycle

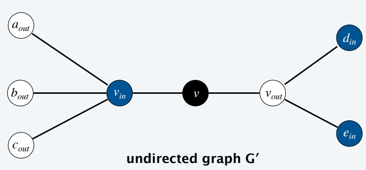
DIRECTED-HAMILTON-CYCLE. Given a directed graph $G = (V, E)$, does there exist a directed cycle Γ that visits every vertex exactly once?

Theorem. DIRECTED-HAMILTON-CYCLE $\le_P$ HAMILTON-CYCLE.

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



directed graph G



undirected graph G′

3-satisfiability reduces to directed Hamiltonian cycle

Theorem. 3-SAT $\le_P$ DIRECTED-HAMILTON-CYCLE.

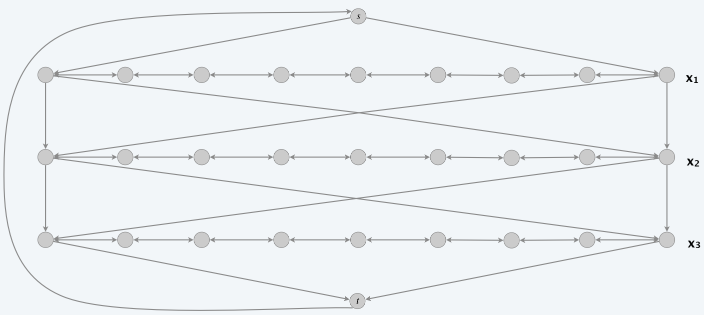
Pf. Given an instance Φ of 3-SAT, we construct an instance G of DIRECTED-HAMILTON-CYCLE that has a Hamiltonian cycle iff Φ is satisfiable.

Construction overview. Let n denote the number of variables in Φ. We will construct a graph G that has 2^n Hamiltonian cycles, with each cycle corresponding to one of the 2^n possible truth assignments.

3-satisfiability reduces to directed Hamiltonian cycle

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

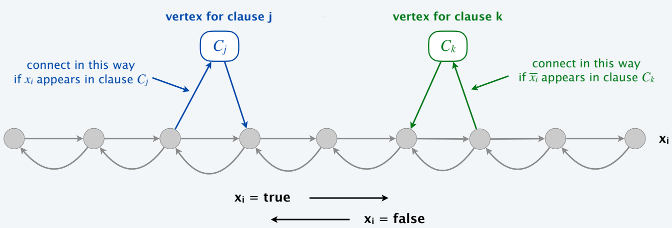
- Construct G to have 2^n Hamiltonian cycles.
- Intuition: traversing path i from left to right $\Leftrightarrow$ setting variable $x_i = true$.



3-satisfiability reduces to directed Hamiltonian cycle

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

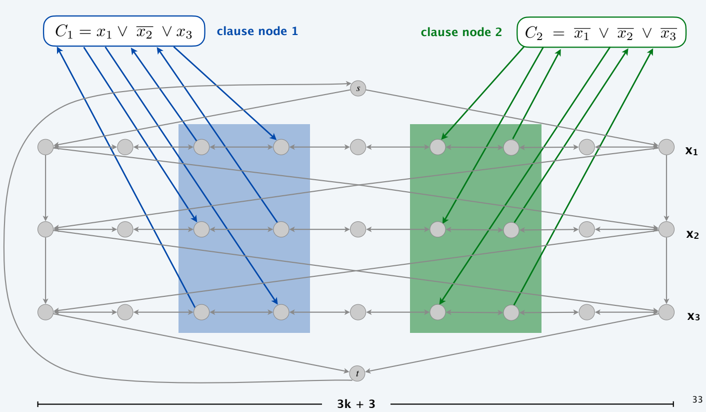
- For each clause: add a vertex and 2 edges per literal.



3-satisfiability reduces to directed Hamiltonian cycle

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

- For each clause: add a vertex and 2 edges per literal.

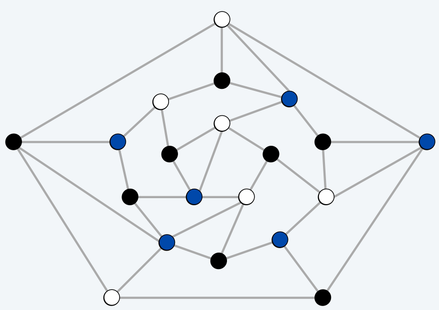


CSCI 355: Algorithm Design and Analysis
10. Intractability

- poly-time reductions
- packing and covering problems
- constraint satisfaction problems
- sequencing problems
- graph colouring
- numerical problems

3-colourability

3-Colour. Given an undirected graph G, can the vertices be coloured black, white, and blue so that no adjacent vertices have the same colour?



yes instance

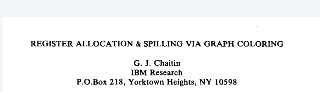
Application: register allocation

Register allocation. Assign program variables to machine registers 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. Vertices are program variables; there exists an edge between u and v if there exists an operation where both u and v are "live" at the same time.

Observation. [Chaitin 1982] We can solve the register allocation problem iff the corresponding interference graph is k-colourable.

Fact. 3-Colour $\leq_P$ K-Register-Allocation for any constant $k \geq 3$.



REGISTER ALLOCATION & SPILLING VIA GRAPH COLORING

G. J. Chaitin
IBM Research
P.O.Box 218, Yorktown Heights, NY 10598

3-satisfiability reduces to 3-colourability

Theorem. 3-SAT $\leq_P$ 3-COLOUR.

Pf. Given a 3-SAT instance Φ, we construct an instance of 3-COLOUR that is 3-colourable iff Φ is satisfiable.

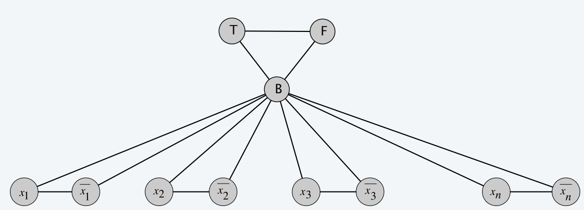
38

3-satisfiability reduces to 3-colourability

Construction.

(i) Create a graph G with a vertex for each literal.
(ii) Connect each literal to its negation.
(iii) Create 3 new vertices 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 vertices and 13 edges.

a small partial instance of one problem
that simulates a certain aspect of the other problem



39

CSCI 355: ALGORITHM DESIGN AND ANALYSIS
10. INTRACTABILITY

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

Subset sum

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

Ex. $\{ 215, 215, 275, 275, 355, 355, 420, 420, 580, 580, 655, 655 \}$, $W = 1505$.

Yes! $215 + 355 + 355 + 580 = 1505$.

Remark. With arithmetic problems, input integers are encoded in binary. Poly-time reduction must be polynomial in binary encoding.

3-satisfiability reduces to subset sum

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

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

3-satisfiability reduces to subset sum

Construction. Given a 3-SAT instance Φ with n variables and k clauses, we form $2n + 2k$ decimal integers, each having $n + k$ digits:

- Include one digit for each variable x_i and one digit 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	=	$\neg x_1$	∨	x_2	∨	x_3
C_2	=	x_1	∨	$\neg x_2$	∨	x_3
C_3	=	$\neg x_1$	∨	$\neg x_2$	∨	$\neg x_3$

3-SAT instance

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

SUBSET-SUM instance

Knapsack

KNAPSACK. Given $2n$ natural numbers $w_1, \ldots, w_n, v_1, \ldots, v_n$ and an integer W, is there a subset that maximizes the v_i while adding up the values w_i to exactly W?

Remark. The knapsack problem is essentially the subset sum problem, but with values in addition to weights.

Subset sum reduces to knapsack

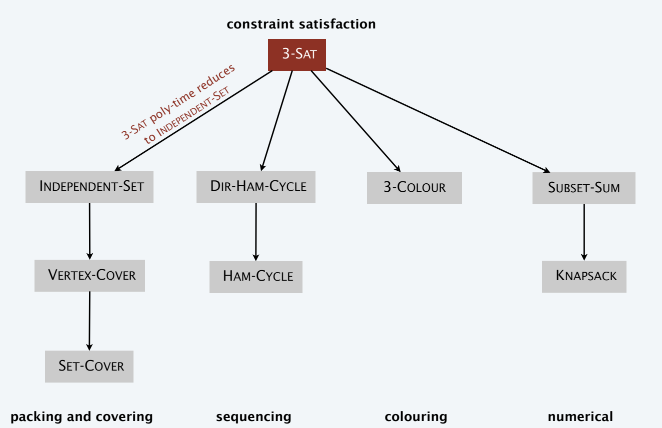
Theorem. SUBSET-SUM $\leq_P$ KNAPSACK.

Pf. Given an instance Φ of SUBSET-SUM, we construct an instance of KNAPSACK that has a solution iff Φ has a solution.

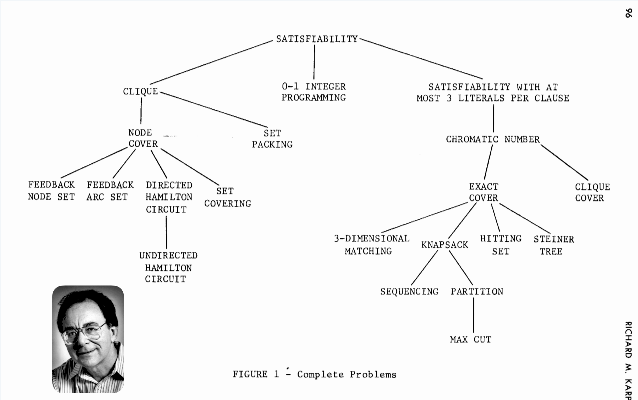
Construction.

- Set each value v_i to be equal to w_i.
- Take the "goal" value to be equal to W.

Poly-time reductions



Karp's 20 poly-time reductions from satisfiability



Dick Karp (1972)
1985 Turing Award