COMS4236: Introduction to Computational Complexity

Spring 2018

Mihalis Yannakakis

Lecture 16, 3/8/18

Outline

- Problems with numbers
  - strong vs. weak NP-hardness
  - pseudopolynomial algorithm
- coNP
- NP$\cap$coNP
- Factoring

Subset Sum

- Input: set S of (positive) integers, another integer t
- Question: ∃ subset T of S that sums to t?
- Note: numbers given in binary
- There is a pseudopolynomial algorithm: runs in time polynomial in the *value* of the numbers (not the bit-size)

- A problem is called strongly NP-complete if it is NP-complete even if the numbers are given in unary notation instead of binary.
- Subset Sum is not strongly NP-complete.
- It is weakly NP-complete

Subset Sum

- Input: set S of (positive) integers, another integer t
- Question: ∃ subset T of S that sums to t?
- Note: numbers given in binary

- In NP: certificate = subset T
- NP-hard: Reduction from Node Cover
- Given graph G=(N,E), bound k for Node Cover → instance of Subset Sum where S has one integer a_i for every node i of G, and one integer b_{ij} for every edge (i,j) of G.
- If G has e edges, then each integer has 2e+1 bits

$$\text{Target number } t = k \cdot 4^e + \sum_{i=0}^{e-1} 2 \cdot 4^i$$

Node Cover $\leq_{log}$ Subset Sum

2e+1 bits: leading bit + 2 bits per edge (edges in any order)

Leading bit= 1 for node-numbers a_i, 0 for edge-numbers b_{ij}

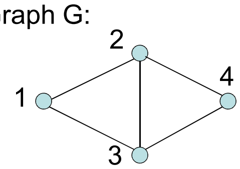
		edge bits:
a_i:	1 -- -- -- 00 01 -- --	01 if $i \in$ edge, 00 otherwise
b_{ij}:	0 00 .. .. 00 01 ….	All 0 except 01 at edge (i,j)

t: bin(k) 10 10 10 10 10 — Target t = k in binary followed by 10 for all edges

• For any subset T, when we add up the numbers in T, there is no carry from bits of one edge to the next or to the leading bit, because only three numbers have a 1 in the 2 bits for an edge

Example

Graph G:



k =3

edges

		(1,2)	(1,3)	(2,3)	(2,4)	(3,4)
a_1:	1	0 1	0 1	0 0	0 0	0 0
a_2:	1	0 1	0 0	0 1	0 1	0 0
a_3:	1	0 0	0 1	0 1	0 0	0 1
a_4:	1	0 0	0 0	0 0	0 1	0 1
b_{12}:	0	0 1	0 0	0 0	0 0	0 0
b_{13}:	0	0 0	0 1	0 0	0 0	0 0
b_{23}:	0	0 0	0 0	0 1	0 0	0 0
b_{24}:	0	0 0	0 0	0 0	0 1	0 0
b_{34}:	0	0 0	0 0	0 0	0 0	0 1
Target number t:	1 1	1 0	1 0	1 0	1 0	1 0

6

Node Cover $\leq_{log}$ Subset Sum

- If $\exists$ node cover C with k nodes then $\exists$ subset T of S that sums to t
- $T = \{ a_i \mid i \in C \} \cup \{ b_{ij} \mid i \notin C \text{ or } j \notin C \}$ sums to t

- If $\exists$ subset T of S that sums to t then $\exists$ node cover C with k nodes
- Because there is no carry from bits of one edge to the next and t has 10 for all edges, T must contain for each edge (i,j) at least one of a_i, a_j
- Because of the k in the leading bits of t, T must contain exactly k numbers a_i $\Rightarrow C = \{ i \mid a_i \in T\}$ is a node cover of size k.

0-1 Knapsack

Input: integers $v_1,\ldots,v_n$, $w_1,\ldots,w_n$ (values, weights of the items), W (knapsack capacity), bound b on total value

Question: $\exists$ subset $C \subseteq \{1,\ldots,n\}$ s.t. $w(C) \leq W$ and $v(C) \geq b$?

Reduction from Subset Sum

Instance $S=\{s_1,\ldots,s_n\}$, t of Subset Sum $\rightarrow$ instance of 0-1 Knapsack: n items, $v_i = w_i = s_i$, knapsack capacity W=t, value bound b=t

- $\exists$ subset T of S that sums to t iff $\exists$ subset $C \subseteq \{1,\ldots,n\}$ s.t. $w(C) \leq W$ and $v(C) \geq b$

0-1 Integer Linear Inequalities

- Input: Set of linear inequalities
- Question: Is there a 0-1 assignment to the variables that satisfies the inequalities?

- In NP: Certificate = satisfying assignment
- Integer Linear Inequalities (∃ integer solution? not only 0-1) also in NP: if there is an integer solution to a set of linear inequalities, then there is one of size (#bits) polynomial in the input size (not trivial to show)

- NP-complete
- Subset Sum: Is there 0-1 solution to $s_1x_1 + \ldots s_nx_n = t$?
- But even strongly NP-complete.

Node Cover $\leq_{log}$ 0-1 Integer Linear Inequalities

- Given graph G and bound k construct instance of 0-1 ILE
  - one variable x_i for each node i of G
  - inequalities $x_i + x_j \geq 1$ for each edge (i,j) of G
  - inequality $\Sigma x_i \leq k$

- 1-1 correspondence between subsets of nodes and 0-1 assignments (=characteristic vectors of subsets)
- A subset covers all the edges and has size $\leq$ k iff the corresponding 0-1 assignment satisfies all the inequalities

- Corollary: Integer Linear Inequalities also NP-complete
- Proof: Add the inequalities $x_i \geq 0$ and $x_i \leq 1$ for all i

Class coNP

- Definition of NP is nonsymmetric with respect to Yes, No

$$\text{coNP} = \{ L \mid \bar{L} = \Sigma^* - L \in NP \}$$

- A decision problem Π is in coNP if the complement of its Yes language L_Π is in NP $\Leftrightarrow$ its No language (set of No instances) is in NP

Certificate version

A language L is in coNP if there is a polynomial-time decidable binary predicate R(.,.) and constant c such that

$L = \{ x \mid \forall y.\ |y| \le |x|^c \text{ predicate } R(x,y) \text{ is true } \}$

coNP-complete problems

- Complements of NP-complete problems
- UNSAT: Given Boolean formula, is it unsatisfiable?
- TAUTOLOGY (VALIDITY): Given Boolean formula, is it a tautology (valid), i.e. satisfied by all truth assignments?
- NONHAMILTONICITY: Given a (undirected or directed) graph, is it nonHamiltonian?
- NON 3-COLORABILITY: Given an undirected graph, is it the case that it has no 3-coloring?
- NODE COVER LOWER BOUND: Given graph G and number k, does every node cover of G have ≥k nodes?
- INDEPENDENT SET UPPER BOUND: Given a graph G and number k, does every independent set of G have ≤k nodes?

Properties

•P⊆NP, P⊆coNP , thus, P⊆NP∩coNP

• NP is closed under union, intersection

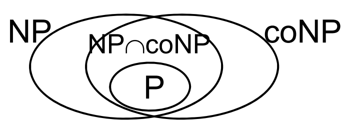
• coNP is also closed under union, intersection

• NP (and coNP) closed under complement iff NP=coNP

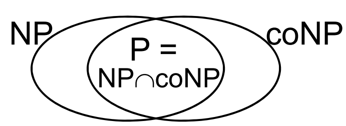
- conjectured not

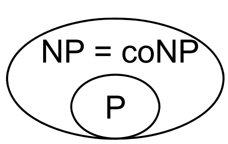
Relations between P, NP, coNP

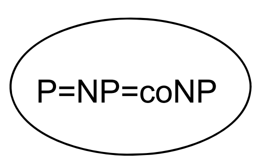
Conjectured relation: all distinct



Other possibilities:







Fundamental Questions

- P = NP?

Is it always as easy to generate a proof as it is to check a proof that is given to us?

- NP = coNP?

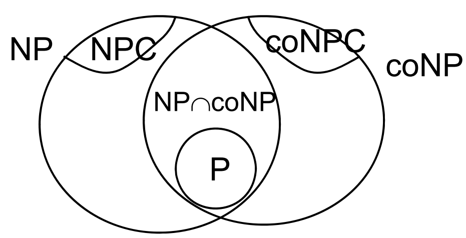
Is it always possible to provide simple convincing evidence that something does not exist, as it is to show that something exists by exhibiting it?

- P = NP∩coNP?

If there is a simple convincing proof both for the presence and the absence of a property, does this mean we can test the property efficiently?

Relations between P, NP, coNP and their complete problems

Conjectured relation:
all distinct



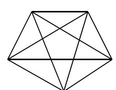
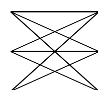
- If an NP-complete problem is in P then P=NP
- If an NP-complete problem is in coNP then NP=coNP

⇒ If NP≠coNP then no NP-complete problem can be in coNP

- equivalently, no problem in NP∩coNP can be NP-complete

NP∩coNP

- Short, easy to check certificates both for the Yes and the No instances
- Examples:
- Graph Bipartiteness:
  - bipartite ⇔ nodes can be partitioned into two sets V1, V2 so that all edges connect a node in V1 with a node in V2
  - nonbipartite ⇔ there is an odd length cycle
- Graph Planarity
  - planar ⇔ can draw on the plane so that no edges intersect
  - nonplanar ⇔ contains a homeomorph of K_5 or K_{33} (Kuratowski's theorem)

K_5  K_{33} 

- These particular properties happen to be in fact in P

NP∩coNP and Optimization

- Optimization problems whose decision version is in NP: assume solutions have polynomial size (#bits) and cost (or values) can be computed in polynomial time
- If there is a polynomial time (or even NP) optimality testing algorithm, which, given an instance and a solution, tests that the solution is optimal, then the decision version is in coNP, and hence probably not NP-complete.
- Proof: Given an instance x and a bound k, guess a solution s, verify that it is optimal, and verify that its cost (or value) is worse than k (i.e. cost(s) >k for a minimization problem, or value(s)<k for maximization).

- Examples: Linear Programming
- Maximum flow problem
- Maximum matching ….

These particular problems turn out to be in fact in P

Primality

- Input: Positive integer N (given in binary: input size n=logN)
- Question: Is N prime?

- The straightforward algorithm (try out all numbers < N to see if any divides N) is pseudopolynomial, *not* polynomial

- Primes $\in$ coNP, i.e. Composites $\in$ NP: guess a factor q and verify that q divides N

- Primes $\in$ NP. Not as obvious [Pratt 1975]

- Primes $\in$ P. Much harder [Agrawal, Kayal, Saxena 2002]

Factoring

- Input: Positive number N
- Output: The prime factorization of N, $N = p_1^{i_1} p_2^{i_2} \cdots p_k^{i_k}$
- Most common cryptographic scheme (RSA) based on presumed difficulty of the factoring problem

- Decision version

Input: Number N, number b

Question: Does N have a (nontrivial) factor $\leq$ b?

(Note: N has a factor $\leq$ b iff it has a prime factor $\leq$ b)

- Can factor N with polynomially many calls to an algorithm for the decision problem: Use binary search to find the smallest (nontrivial) factor of N (it will be a prime), divide N by the factor, and repeat.

Factoring Decision ∈ NP∩coNP

- Decision ∈NP: Guess a factor $p > 1$ and $\leq$ b, and check that p divides N
- Decision ∈coNP: Guess the prime factorization of N:
  $N = p_1^{i_1} p_2^{i_2} \cdots p_k^{i_k}$, verify the factoring and primality of the p_i
  (can use the primality algorithm in P or the NP algorithm: guess certificates C(p_i) for all the p_i and verify them)
  Check that no p_i is $\leq$ b

- If NP∩coNP=P then Factoring Decision ∈P ⇒ Factoring ∈P
- In other words, Factoring ∉P ⇒ NP∩coNP≠P

- Factoring can be done in polynomial time in a Quantum Computer model [Shor 1994] – more powerful than ordinary computers?

Hardness of Factoring ?

- Probably cannot use NP-hardness to argue that Factoring is an intractable problem:
- If Factoring is NP-hard then NP=coNP
- This holds even for a more general notion of polynomial reduction (and NP-hardness) called *Cook reduction.*
- Problem A *Cook reduces* to problem B if there is an algorithm for problem A that uses a subroutine for B and which runs in polynomial time apart from the subroutine calls Thus if the subroutine for B is polynomial-time then the algorithm A also polynomial-time.
- If SAT Cook reduces to Factoring then can get NP algorithm for UNSAT (a coNP-complete problem): replace the subroutine calls with an NP algorithm that guesses and verifies the factorization.

NP∩coNP and Completeness

- Factoring $\notin$P $\Rightarrow$ NP∩coNP$\neq$P

- Does the converse hold?

  Can we argue that Factoring is complete for NP∩coNP?

- No complete problems known for NP∩coNP
- Basic Obstacle: NP∩coNP is a "semantic class".
- No effective syntactic characterization in terms of a class of machines, as we had with deterministic and nondeterministic time- and space-complexity classes, where we could just limit the amount of time or space used by a TM