Computational Intractability 2010/4/15

Lecture 2

Professor: David Avis *Scribe:Naoki Hatta*

1 P and NP

1.1 Definition of P and NP

Decision problem it requires yes/no answer.
Example:

- X is a set of strings.
- Instance: string s.
- Algorithm A solves problem X: A(s)= 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(.) is some polynomial and $|s|$ is length of s

Certifier 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)=yes.

intuition about Certification algorithm

- Certifier doesn't determine whether s $\in$ X on its own.
- Certifier checks a proposed proof t that s $\in$ X.

Certificate Instance which leads the true answer.

P Decision problems for which there is a poly-time algorithm.

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

Remark NP stands for nondeterministic polynomial-time.

"yes"is easy to check but "no" is not!!!

1.2 Examples

Example 1: Composite

Composites Given integer s, is s composite?

Certificate A nontrivial factor t of s. Note that such a certificate exists iff s is composite.Moreover $|t| \leq |s|$.

Certifier C(s,t) that if s is a multiple of t, return true, but otherwise return false.

Instance s = 437,669

Certificate t = 541 or 809 (437,669 = 541 * 809)

Therefore, Composite is in NP.

Example 2: Satisfiability

3-SAT Given a CNF formula Φ, is there a satisfying assignment?

Certificate An assignment of truth values to the n boolean variables.

Certifier Check that each clause in Φ has at least one true literal.

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

Certificate $x_1 = 1$, $x_2 = 1$, $x_3 = 0$, $x_4 = 1$

Therefore, 3-SAT is in NP.

Example 3: Hamiltonian Cycle

HAM-CYCLE Given an undirected graph G=(V,E), does there exist a simple cycle C that visits every node?

Certificate A permutation of the n nodes.

Certifier Check that the permutation contains each node in V exactly once, and that there is an edge between each pair of adjacent nodes in the permutation.

Instance & Certificate as figure1

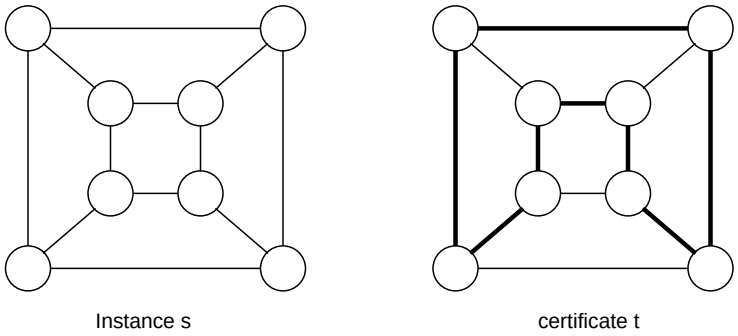

Figure 1: HAM-CYCLE

Therefore, HAM-CYCLE is in NP.

1.3 P vs NP

P Decision problems for which there exists a **poly-time algorithm**.

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

EXP Decision problems for which there exists an **exponential-time algorithm** .

Claim P $\subseteq$ NP.

Proof Consider any problem X in P.

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

Claim NP $\subseteq$ EXP.

Proof 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 $|t| \leq p(|s|)$.
- Return yes, if C(s,t) returns yes for any of these.

1.3.1 Does P = NP?

Is the decision problem as easy as the certification problem?

If yes Efficient algorithms for 3-COLOR,TSP,FACTOR,SAT,...

If no No efficient algorithms possible for 3-COLOR,TSP,SAT,...

Consensus opinion is P $\neq$ NP!

2 NP-Completeness

2.1 Transformation

Def Problem X **polynomial reduces**(Cook) 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.

Def Problem X **polynomial transforms**(Karp) to problem Y if given any input x to X, we can construct an input y such that x is a yes instance of X iff y is a yes instance of Y.

Note Polynomial transformation is polynomial reduction with just one call to oracle for Y, exactly at the end of the algorithm for X. Almost all previous reductions were of this form. as figure2.

Are these two concepts the same?

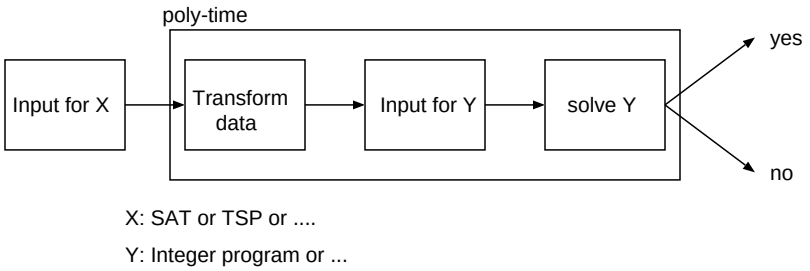

Figure 2: Transformation

2.2 definition

NP-complete A problem Y in NP with the property that for every problem X in NP, X $\leq_p$ Y. ($X \leq_p Y$ means that problem Y can reduce to problem X in poly-time.)

Theorem Suppose Y is an NP-complete problem. Then Y is solvable in poly-time iff P=NP.

Proof($\Leftarrow$) If P = NP then Y can be solved in poly-time since Y is in NP.

Proof($\Rightarrow$) Suppose Y can be solved in poly-time. Let X be any problem in NP. Since X $\leq_p$ Y, we can solve X in poly-time. This implies NP $\subseteq$ P. We already know P $\subseteq$ NP. Thus P = NP.

Do there exist "natural" NP-complete problems?
The "First" NP-complete problem is SAT.

Remark Once we establish first "natural" NP-complete problem, others fall like dominoes.

Recipe to establish NP-completeness of problem Y **step1** Show that Y is in NP.

step2 Choose an NP-complete problem X.

step3 Prove that X $\leq_p$ Y

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

Proof Let W be any problem in NP. Then W $\leq_p$ X $\leq_p$ Y. By transitivity, W $\leq_p$ Y. Hence Y is NP-complete.

Observation All problems in figure3 are NP-complete and polynomial reduce to one another!

NP-hard A problem Y is **NP-hard** if $X \leq_p Y$ for an NP-complete problem X.

Note A decision problem such that every problem in NP reduces to it. Not necessarily in NP.

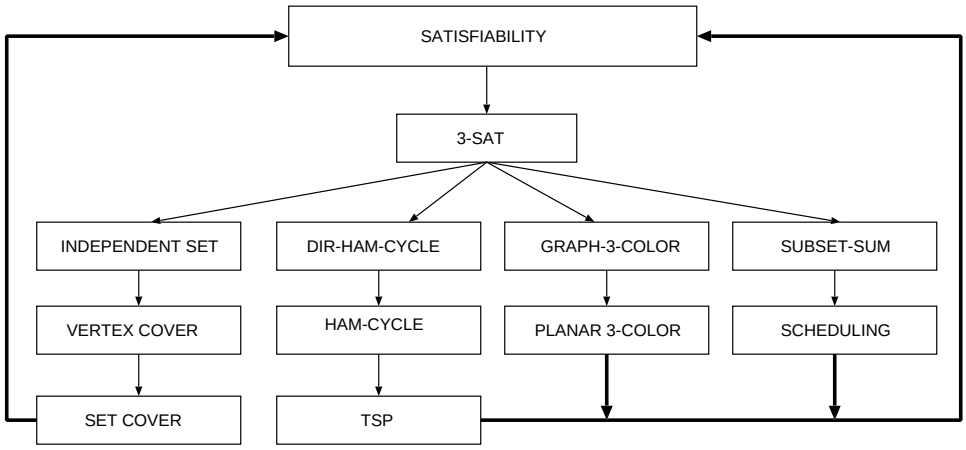

Figure 3: Reduction Graph

2.3 Reduction example

2.3.1 Directed Hamiltonian cycle

3-SAT Reduces to Directed Hamiltonian Cycle

Claim 3-SAT $\leq_p$ DIR-HAM-CYCLE

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

1. create graph G that has 2^n Hamiltonian cycles which correspond in a natural way to 2^n possible truth assignments.
2. regard a clause as a node and add it and 6 edges to the graph G for each clause.

In this way, the new graph is gained. For example,in case that 1 clause and 3 variables, the graph is like figure4.

Intuition traverse path i from left to right $\Leftrightarrow$ set variable $x_i = 1$.

Claim Φ is satisfiable iff G has a Hamiltonian cycle.

Proof($\Rightarrow$) 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

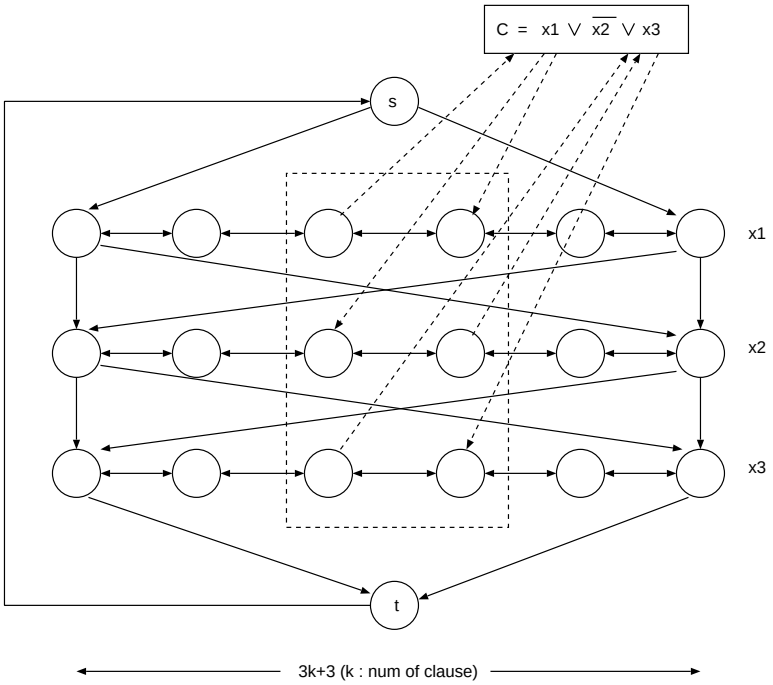

Figure 4: 3SAT to DHC

Proof(⇐) Suppose G has a Hamiltonian cycle Γ.
if Γ enters clause node C_j,

1. it must depart on mate edge.(Thus, nodes immediately before and after C_j are connected by an edge e in G.)
2. 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 $G - \{C_1, C_2, ..., C_k\}$. And then, determine assignment from Γ'.

- Set $x*_i = 1$ iff Γ' traverses row i left to right.
- Set $x*_i = 0$ iff Γ' traverses row i right to left.

Since Γ' visits each clause node C_j, at least one of the paths is traversed in "correct" direction, and each clause is satisfied.

2.3.2 SUBSET-SUM

Given natural numbers $w_1, w_2, ..., w_n$ and an integer W, is there a subset that adds up to exactly W?

Ex: $\{1, 4, 16, 64, 256, 1040, 1041, 1093, 1284, 1344\}$, $W = 3754$.

Ans: 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 $\leq_p$ SUBSET-SUM

Proof 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.

For example, the instance of 3-SAT.

- $C_1 = \overline{x} \vee y \vee z$
- $C_2 = x \vee \overline{y} \vee z$
- $C_2 = \overline{x} \vee \overline{y} \vee \overline{z}$

These 3 CNF transform into the instance of SUBSET-SUM, as illustrated table1.

	x	y	z	C_1	C_2	C_3	w_i
x	1	0	0	0	1	0	100,110
¬x	1	0	0	1	0	1	100,001
y	0	1	0	1	0	0	10,000
¬y	0	1	0	0	1	1	100,001
z	0	0	1	1	1	0	1,010
¬z	0	0	1	0	0	1	1,101
	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

Table 1: 3SAT to SUBSET-SUM

Last 2k rows are the dummies to get clause columns to sum to 4, because that the sum of a $C_j$column equals 4 means that the clause can be satisfiable. Each row correspond the instance (natural number) of SUBSET-SUM, and iff there exists a subset that sums to exactly $W(= 111,444)$, Φ is satisfiable.

References

[1] Jon Kleinberg and Eva Tardos, *Algorithm Design* (ADDISON WESLEY, 2005)