E0 224 Computational Complexity Theory **Indian Institute of Science, Bangalore**

Fall 2014 **Department of Computer Science and Automation**

Lecture 5: Aug 20, 2014

Lecturer: Chandan Saha <chandan@csa.iisc.ernet.in> *Scribe: Jaiprakash*

1. Reductions

Definition 1.

SAT (Boolean satisfiable) problem: SAT problem for a given Boolean formulae tries to answer, whether there exists a truth assignment making the Boolean formula true. In other words, can we assign the variables of a given Boolean formula such a way as to make the formula evaluate to true.

Cook-Levin Theorem: Cook-Levin Theorem states that SAT problem is NP-complete. So there exist a poly-time computable function f s.t $\{x \in L$ iff $f(x) = \phi_x$ is satisfiable$\}$, where L is the language in NP i.e. $L \leq_p SAT$

Definition 2.

Levin-reduction: The above $L \leq_p SAT$ reduction not only satisfies $\{x \in L$ iff ϕ_x is satisfiable$\}$, but it also provides an efficient way of transforming a certificate for x to a satisfying assignment for ϕ_x and vice versa.

Observation 1.

It is easy (poly-time) to find a certificate for a x (where $x \in L$) from a satisfying assignment of ϕ_x. Such reductions are known as Levin reduction.

Observation 2.

There is a one-to-one onto map between certificates of x and certificates of ϕ_x.

Suppose u is a certificate of x

$\Psi_x : u \longrightarrow (u, g(x, u))$ and $u \longleftarrow (u, z)$

where, z is the snapshot of the Turing machine.

It provides a one-to-one and onto map between the set of certificates for x and the set of satisfying assignments for ϕ_x, (So they are of same size)

Definition 3.

Parsimonious reductions: Reductions satisfying Observation 2 are known as Parsimonious reductions.

Cook-Levin(Restated): For every language $L \in NP$, there is a parsimonious reduction from L to SAT.

Definition 4.

Running Time of a TM: Let $T : N \rightarrow N$ and $f : \{0,1\}^* \rightarrow \{0,1\}^*$. We say that a TM M computes f in time $T(n)$ iff on every input $x \in \{0,1\}^*$, $M(x) = f(x)$ and number of basic operations(number of times M applies a rule from the truth table) of M on input x is bounded by $T(|x|)$.

Fact: ϕ_x is of size $O(T(|x|) \log T(|x|))$

2. Independent/Stable set

Definition 5.

Independent set of a graph G is a set of vertices of G such that no two of which are adjacent in G.

The problem of finding existence of independent set of size k can be formulated as

$$INDSET = \{< G, k >: \exists\ S \subseteq V(G) \text{ s.t. } |S| = k \text{ and } \forall u, v \in S, (u, v) \notin E(G)\}$$

Theorem 1.

INDSET problem is NP-Complete

Proof. To show that INDSET problem is NP-Complete, we have to show INDSET is both NP and NP-Hard.

1. **INDSET is NP.**

   Suppose somebody has given a certificate that contains a set of vertices, in polynomial time we can verify whether the number of vertices given in the certificate is equal to k and also verify whether two vertices are not adjacent.

2. **INDSET is NP-Hard.**

   We will prove this by doing a polynomial time reduction from 3-SAT to INDSET(i.e 3-SAT $\leq_p$ INDSET).

   Suppose ϕ is in 3-CNF. Our goal is to define poly time computable map

   f: $\phi \longrightarrow f(\phi) =< G, k >_\phi$ s.t $\phi \in$ 3-SAT iff $< G, k >_\phi \in INDSET$

   Let m be the number of clauses in ϕ. The graph G is defined as follows: we associate a cluster of 7 vertices in G with each clause of ϕ. The vertices in a cluster associated with a clause C, corresponds to the 7 possible satisfying partial assignments to the three variables on which C depends (If C depends on less than three variables, then we repeat one of the partial assignment).Inside a cluster we connect all pair of vertices and put an edge between two vertices of G if they correspond to inconsistent partial assignments. The below figure shows the transformation.

   The transformation from ϕ to G can be done in polynomial time.

We are now showing that ϕ is satisfiable iff G has an independent set of size m.

Case 1: Suppose ϕ has a satisfying assignment u. We are going to define $S \subseteq G(V)$ of size m. For each clause C of ϕ, add a vertex in S that correspond to the restrictions of the assignment u to the variables C depends. Since

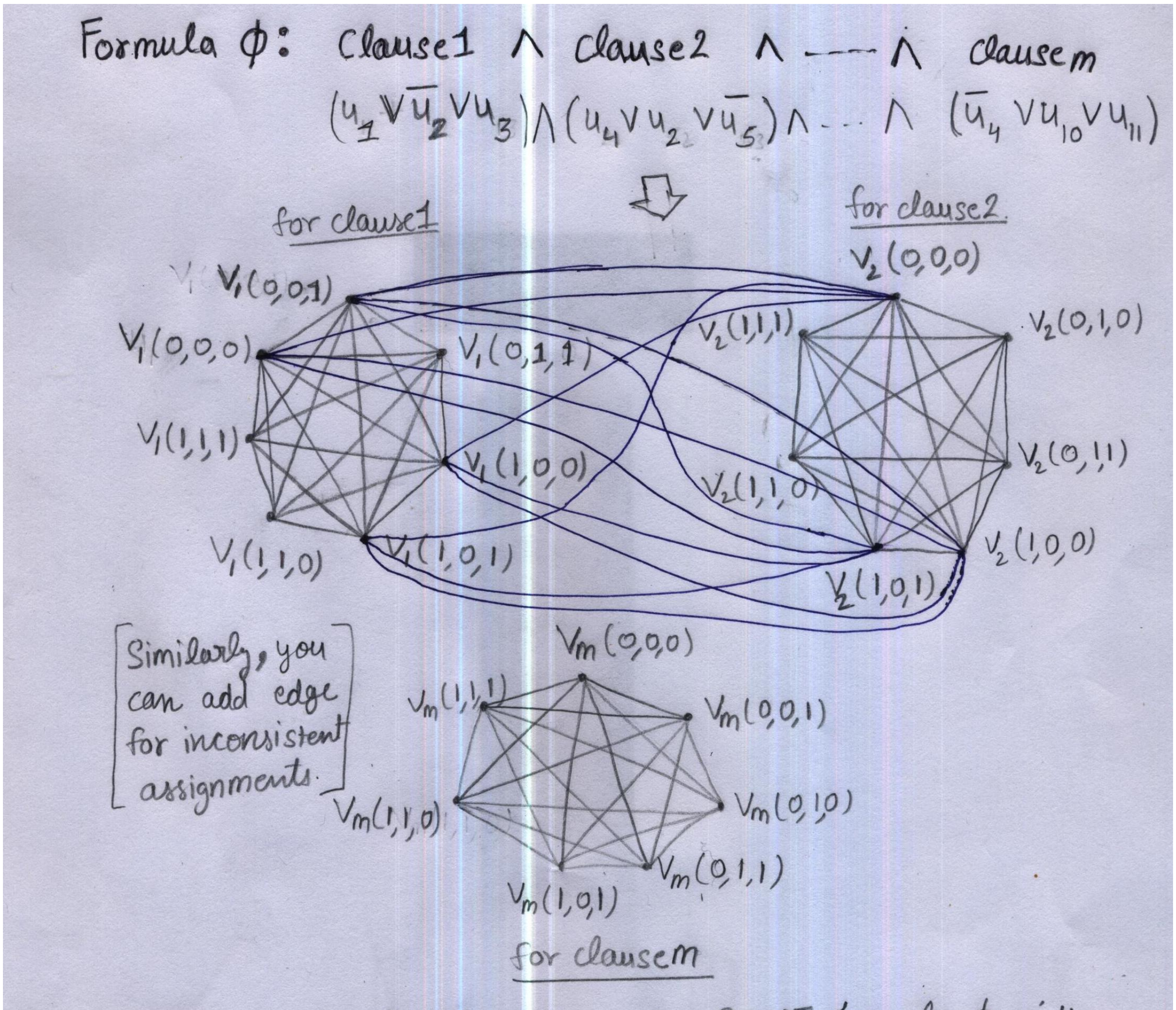

Fig: Example showing transformation of a 3CNF formula ϕ with 'm' clauses into a graph G with 7m vertices.

no two vertices of S correspond to inconsistent assignments. Hence S is an independent set of size m.

Case 2: Suppose G has an independent set of size m. We are going to define assignment u of ϕ. For every $i \in [n]$(n is the number of variables in ϕ), if there is a vertex in S, whose partial assignment gives a value a to u_i, then set $u_i = a$, otherwise $u_i = 0$. Since S is an independent set, each variable u_i can take at most one value. Since we put all the edges within each cluster, S can have at most a single vertex from each cluster. So $|S| = m$, implies S has exactly one vertex from every cluster which is the satisfying assignment. Hence it satisfies all of ϕ clauses. $\square$

Homework 1.

Prove or disprove that the above reduction is Parsimonious Reduction.

3. 0/1 Integer Programming

Definition 6.
0/1 IP is a mathematical feasibility(optimization) program in which variables can take only 0 or 1 values.
Input: set of linear constraints with rational coefficients.

$$\mathbf{A}\text{x} \leq \text{b s.t x} \in \{0,1\}$$

Task: Check whether there is a 0/1 assignment to the x-variables s.t. all the given constraints are satisfied.

Theorem 2.
0/1 Integer Programming is NP-Complete.

Proof. To prove this we have to show,

1. **0/1 IP is NP.**
   If somebody gives a certificate that contains 0/1 assignment of variables, then we can verify whether provided certificate is satisfiable or not, in polynomial time.

2. **0/1 IP is NP-Hard.**
   We will prove this by doing a polynomial time reduction from 3-SAT to 0/1 IP (i.e 3-SAT $\leq_p$ 0/1 IP).
   Let $\phi = C_1 \wedge C_2 \wedge .... \wedge C_m$, and let the variables in the 3-SAT formula be $x_1, x_2, ..., x_n$ and their corresponding variables $z_1, z_2, ..., z_n$ in our 0/1 IP.
   For each clause C_i(e.g. $x_1 \vee x_2 \vee \neg x_3$) we have a constraint (like: $z_1 + z_2 + (1 - z_3) \geq 1$). To satisfy this inequality we must set $z_1 = 1$ or $z_2 = 1$ or $z_3 = 0$, which means $x_1 = true$ or $x_2 = true$ or $x_3 = false$ in the corresponding truth assignment.

□

4. Remark on Factoring

FACT$_1$:= $\{< N, L, U >$: if there exist a number $\in [L, U]$ that divides $N\}$
The FACT$_1$ problem is NP-Complete.

FACT$_2$:= $\{< N, L, U >$: if there is a prime p in the interval $[L, U]$ that divides N $\}$
Note: We can factor a number using $O(\sqrt{N})$ trials division, but the representation of N have $\log N$ bits, hence it is an exponential time algorithm.
FACT$_2$ is in NP. The current best algorithm for FACT$_2$ runs in time $2^{O((\log N)^{(1/3)}(\log \log N)^{2/3})}$.

5. Subset Sum Problem

Definition 7.
Given a (multi)set X of integers and an integer k, does there exist a non-empty subset of X whose sum is k.

SUBSET-SUM= $\{< X, k > | X = \{x_1, x_2, ..., x_n\}$ and $\exists\ A \subseteq [n]$ s.t $\sum_{i \in A} x_i = k\}$
Input: Given $X = \{x_1, x_2, ..., x_n\}$ and k
Task: Check if $\exists\ A \subseteq [n]$ s.t $\sum_{i \in A} x_i = k$

Theorem 3.
SUBSET-SUM Problem is NP-complete.

Proof. To prove this we have to show,

1. **SUBSET-SUM Problem is NP.**
   If somebody gives a certificate that contains a subset of X, then we can verify whether provided certificate is subset of X and sum up to k in polynomial time.

2. **SUBSET-SUM Problem is NP-Hard.**
   We prove this by showing 3-SAT $\leq_p$ SUBSET-SUM

   (a) Let we have l variables $\{v_1, .., v_i, .., v_l\}$ and m clauses $\{c_1, .., c_j, .., c_m\}$.

   (b) For each variables v_i create a number t_i and f_i of $(l + m)$ digits.

      i. The i^{th} digit of t_i and f_i is equal to 1.

      ii. For all j, $l + 1 \leq j \leq l + m$

         A. $t_{i,j} = 1$, if v_i is in clause c_{j-l}, 0 otherwise.

         B. $f_{i,j} = 1$ if $\neg x_i$ is in clause c_{j-l} , 0 otherwise.

      Example:

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

	i			*j*			
Number	1	2	3	1	2	3	4
t_1	1	0	0	1	0	0	1
f_1	1	0	0	0	1	1	0
t_2	0	1	0	1	0	1	0
f_2	0	1	0	0	1	0	1
t_3	0	0	1	1	1	0	1
f_3	0	0	1	0	0	1	0

   (c) For each clause c_j, create x_j and y_j of length $(l + m)$ and initialize it to 0.

      i. assign $x_{j,l+j}$ and $y_{j,l+j}$ to 1.

   (d) Create sum s, of length $(l + m)$

      i. For j, $1 \leq j \leq n$,$s_j = 1$

      ii. For j, $l + 1 \leq j \leq l + m$,$s_j = 3$

	i			*j*			
Number	1	2	3	1	2	3	4
x_1	0	0	0	1	0	0	0
y_1	0	0	0	1	0	0	0
x_2	0	0	0	0	1	0	0
y_2	0	0	0	0	1	0	0
x_3	0	0	0	0	0	1	0
y_3	0	0	0	0	0	1	0
x_4	0	0	0	0	0	0	1
y_4	0	0	0	0	0	0	1

$\square$

6. Cryptosystem Based on SUBSET-SUM Problem.

Merkle-Hellman is a public/asymmetric key encryption proposed by **Ralph Merkle and Martin Hellman** in 1978.

Definition 8.
Public Key : It is made available to everyone via a publicly accessible repository or directory.

Definition 9.
Private Key : The Private Key must remain confidential to its respective owner.

Private Key :

1. A superincreasing sequence $e =< e_1, e_2, ..., e_n >$ s.t. $\forall i \in n, e_i > \sum_{j=1}^{i-1} e_j$
2. Pick a random integer m s.t $m > \sum_{i=1}^{n} e_i$.
3. A number w that is relatively prime to m. i.e $\exists w'$ s.t. $w' \cdot w \equiv 1 \mod m$.

Public Key :

A sequence of numbers $(h_1, h_2, ..., h_n)$ s.t. $h_i \equiv w \cdot e_i \mod m$

Encryption :

Let $X = (x_1, x_2, ..., x_n)$ is the plain(original) message which a user wants to send. It is a 0-1 string.

To encrypt n-bit message X, calculate Ciphertext $C = \sum_i h_i \cdot x_i$, which is the encrypted message and is send to the receiver.

Decryption

Receiver receives C. In order to decrypt a Ciphertext C, a receiver has to find the message bits x_i.

Receiver have $C = \sum_{i=1}^{n} h_i \cdot x_i$

$w' \cdot C \equiv \sum_{i=1^n} w' \cdot h_i \cdot x_i$

$\equiv \sum_i w' \cdot w \cdot e_i \cdot x_i \mod m$ [because $h_i \equiv w \cdot e_i \mod m$]

$\equiv \sum_i e_i \cdot x_i \mod m$ [because $w' \cdot w \equiv 1 \mod m$]

$= \sum_i e_i \cdot x_i$ [because of super-increasing sequence]

Receiver knows $\sum_{i=1}^{n} e_i.x_i = D(say)$ using the private key. Now, this problem is in the same form of SUBSET-SUM Problem and it is easy to solve because X is a superincreasing sequence. If numbers in the set are superincreasing then the problem is solvable in polynomial time using simple greedy algorithm (Reference [3]).

1. Take largest element from e, let e_k
2. If $e_k > D$ then $x_k = 0$, otherwise $x_k = 1$.
3. Subtract $(x_k \cdot e_k)$ from D and remove e_k from e and continue to step-1, until you get X.

References

[1] S.ARORA and B.BARAK Computational Complexity: A Modern Approach, Cambridge University Press, 2009

[2] Michael Sipser ”Introduction to Theory of Computation”, Cengage Learning

[3] Shamir, Adi (1984). ”A polynomial-time algorithm for breaking the basic Merkle - Hellman cryptosystem”. Information Theory, IEEE Transactions on 30 (5): 699704