E0 224 Computational Complexity Theory **Aug-Dec 2015**

Lecture 3: Examples of NP-Complete Problems

Lecturer: Chandan Saha *Scribe: Parth Verma and Ananya*

3.1 3SAT is NP-Complete

SAT Problem :- Given a set of clauses $C_1, C_2, \ldots, C_m$ in CNF form, where C_i contains literals from $x_1, x_2, \ldots, x_n$ problem is to check if all clauses are simultaneously satisfiable. Cook-Levin theorem shows that SAT is NP-Complete.

3SAT problem:- Given a set of clauses $C_1, C_2, \ldots, C_m$ in 3CNF from variables $x_1, x_2, \ldots, x_n$ problem is to check if all the clauses are simultaneously satisfiable.

$$3SAT := \{\text{Satisfiable 3CNF formulae }\}$$

*Proof Sketch:*3SAT problem is in NP since any assignment of values to variables can be verified in polynomial time. We show that 3SAT is also NP-Complete by showing a reduction from SAT to 3SAT in polynomial time.

$$SAT \leq_p 3SAT$$

We reduce each clause of the SAT instance containing more than 3 literals into a conjunction of several new clauses in 3SAT instance as follows:

1. Create a new clause for 3SAT instance containing first two literals of corresponding SAT instance clause and OR it with a free variable say z_1.

2. Make new clause in conjunction with previous clause and add the next literal of SAT instance clause and OR it with $\bar{z_1}$ i.e. negation of free variable used in previous clause and OR it with a new free variable say z_2.

3. Repeat step 2 with next clause in conjunction with previous clause containing next literal of original CNF clause OR with $\bar{z_2}$ and further ORed with z_3.Continue this process till last two literals in original SAT instance clause will be left. Last clause will contain negation of free literal of previous clause ORed with last two literals in corresponding clause in SAT instance.

4. Repeat step 1 to 3 for next SAT instance clause and keep the output in conjunction with previous 3SAT instance clauses.

Hence if a particular SAT instance clause contain m literals where $m > 3$ then the reduced 3SAT instance will contain $m - 2$ clauses in conjunction and contain $m - 3$ free variables. So, this reduction is polynomial in time and space.

Reduction Example:- C = $x_1 \vee \bar{x_2} \vee x_3 \vee \bar{x_4} \vee x_5 \rightarrow (x_1 \vee \bar{x_2} \vee z_1) \wedge (x_3 \vee \bar{z_1} \vee z_2) \wedge (\bar{x_4} \vee x_5 \vee \bar{z_2})$

Claim 1 : If the SAT instance is satisfiable then so is the corresponding 3SAT instance.

SAT instance is satisfiable when every clause evaluates to True simultaneously . If a clause in SAT instance is True then at least one literal in it is True. So in the corresponding reduced 3SAT instance the 3CNF clause containing that literal is True and we can make free literals ORed with this literal to zero so that its negation in its neighboring 3CNF clause evaluates to one. Keep this alteration to make all 3CNF clauses evaluate to one.

Claim 2 : If 3SAT instance is satisfiable then so is SAT instance.

Since each 3CNF clauses evaluates to True, so every reduced 3CNF clauses from original 3SAT instance clause evaluates to True. This can happen only if at least one original literal present in SAT instance clause is True because if all original literals have value False then only free variable can not make all 3CNF clause True since free variable are present in alternative negation in neighboring clause , so at least one clause will be False. Thus atleast one original literal is True which will make corresponding SAT instance clause True.

By Claim 1 and Claim 2 we conclude that 3SAT is NP -Complete.

. . .

3.2 Integer programming (IP)

Given a set of m clauses of the form

$$\sum_{i=1}^{n} a_i x_i \leq b \qquad \forall i \in \{1, 2, .., n\}$$

Check if there exist a 0/1 assignment to each x_i for i∈{1,2,..,n} such that all the m constraints are satisfiable.

Proof :- Again we solve it by reduction from 3SAT.

$$3SAT \leq_p IP$$

Firstly problem IP belongs to NP Class since for a given value of $x_1, x_2, ..., x_n$ we can verify if all constraints are satisfied simultaneously in polynomial time. Now we show its NP-Hard by reduction of 3SAT problem into IP.

Reduction : For each clause in 3SAT instance we reduce each clause into inequality constraints as follow :-

1. If the literal in clause is in negation form of variable say $\bar{x_i}$ then add (1-z_i) in the inequality.

2. If the literal say x_i is not in negation form then simply add z_i in the inequality.

3. Thus for example if a particular 3CNF clause is $x_1 + \bar{x_2} + x_3$ then reduced inequality will be $z_1 + (1 - z_2) + z_3 \geq 1$ which is equivalent to $-z_1 - (1 - z_2) - z_3 \leq -1$ i.e for each literal x_i in the 3CNF clause we write in inequality variable z_i and if literal is in negation form then write as $1 - z_i$ where each z_i can take value {0,1} which is equal to value of corresponding boolean variable x_i in 3SAT clause.

4. Convert each inequality of the form $z_1 + z_2 + z_3 \geq 1$ into the standard form of inequality $-z_1 - z_2 - z_3 \leq -1$ which is obtained by multiplying both side of inequality by -1.

Claim 1:- If 3SAT is satisfiable then IP will have solution.

If each clause in 3SAT is True then atleast one literals in each clause will have True assignment. This will add -1 due to the corresponding variable in the corresponding inequality and hence the constrain will be satisfied. Thus if all clauses are True then all inequality will be satisfied.

Claim 2:- If IP has solution then 3SAT is satisfiable.

If each inequality constraints are satisfied then atleast one variable value in each inequality will make corresponding literals in 3SAT instance clause equals to True and hence 3SAT instance will be satisfiable.

From claim 1 and claim 2 we conclude IP is an NP-Hard and it also belong to NP-class hence it is NP-complete.

3.3 Solvability of Quadratic equation (QE)

Input :- Given m quadratic equations containing boolean literals $x_1, x_2, .., x_n$ over $\mathbb{F}_2$. where each equation is of the form $\sum_{i,j\in[n]} x_i x_j = b$, where addition is modulo 2. We show that Quadeq is NP-complete

Task :- Check if they are solvable i.e. if $\exists$ a 0/1 assignment of variables that satisfies all the equations.

Claim :- Quadeq is NP-Complete.

Proof:- Given decision problem belongs to NP-class since for a given value of $x_1, x_2, ..., x_n$ verifier can verify if each equations are satisfied in polynomial time.

We show its NP -Hard by reduction of 3SAT problem to QE.

Reduction :- We reduce each clause in 3SAT instance into equivalent QE in polynomial time as

1. If literal is in non-negation form x_i then reduce it to simply z_i and if it is in negation form $\bar{x_i}$ then reduce it to $(1 - z_i)$ and multiply the reduced term with free variable say a. Thus if variable x_i is True in the clause then corresponding variable z_i in equation has value 1 and if x_i is False then z_i has value 0.

2. Thus if a clause in 3SAT instance is of the form $x_1 \vee \bar{x_2} \vee x_3$ then reduced QE will be $az_1 + b(1-z_2) + cx_3 = 1$ mod 2. Where a,b,c are free boolean variable that can be manipulated to satisfy the equation and its use is restricted to this equation only.

3. Thus corresponding to m clauses in 3SAT instance we will have m QEs.

Claim 1 :- If 3SAT instance is satisfiable then all QEs will be satisfied.

If a clause in 3SAT instance is True then at least one literal in it is True, this literal will contribute value one to the reduced form and free variable multiplied with it can have value one and remaining free variable can have value zero to equate this equation to one. Same thing we can do for all QEs to satisfy.

Claim 2 :- If all QEs are satisfied then 3SAT instance is satisfiable.

If a QE equates to value one then at least one reduced variable have the value such that the corresponding literal in 3SAT clause has value True since QEs can always evalutes to one by manipulating free variable except when all the terms multiplied with free variables are all zero. Thus if all QEs are satisfied then all clauses evaluates to True.

Thus by claim 1 and claim 2 we conclude that Quadeq problem is NP-hard and since it also belong to NP-class. Thus it is NP-complete.

. . .

3.4 Independent set

A set of vertices U in a graph is an *independent set* if no two vertices have an edge between them in G. In the INDEPENDENT SET (IS) problem, given a graph G and an integer K, we have to decide if there is an independent set of size K in G.

Claim: IS is NP-Complete

Proof: Given a set, we can check in polynomial time if it is an independent set by checking every pair of vertices in the given set. Therefore, IS is in NP.

We will show that IS is NP-Complete by reducing the 3SAT problem to IS problem in polynomial time.

Transformation from 3SAT to IS problem: For every clause of 3SAT, we construct a complete graph on 7 vertices. The vertices in the graph stand for the 7 possible partial assignments to the three variables of the clause. Doing so for every clause gives us m clusters in the graph G corresponding to m clauses in 3SAT. An edge is put between two vertices in the graph if there is an inconsistency in the two partial assignments corresponding to the vertices. For example, say $x_1 = 0, x_2 = 1, x_3 = 1$ and $\bar{x_1} = 0, x_2 = 1, x_4 = 1$ are the partial assignments. Then, there is an edge between the vertices corresponding to these two assignments because there is an inconsistency in the values assigned to x_1.

3SAT problem with m clauses is satisfiable iff G has an independent set of size m.

($\Rightarrow$) If the 3SAT problem is satisfiable, there exists a consistent partial assignment to the m clauses. This implies that there are m vertices in the m clusters of the graph G with no edges between them. This is true because an edge is put between two vertices only if the assignment is inconsistent. Therefore, we have a independent set with size m.

($\Leftarrow$) If there is an independent set of size m in G, there are m vertices in G that do not have an edge between them. These vertices have to be in different clusters as within a cluster all the vertices are connected. This means there is an assignment which is consistent with all the clauses.

3.5 Vertex cover

A set of vertices U in a graph $G = (V, E)$ is a *vertex cover* if every edge $e \in E$ is incident on at least one vertex in U. In the VERTEX COVER (VC) problem, given a graph G and an integer K, we have to decide if there is a vertex cover S with K vertices.

Claim: VC is NP-Complete

Proof: Given a set of vertices, we can check if it is a vertex cover by looking at every edge $e \in E$ and then, checking if the size of the set is at most K. This can be done in polynomial time. Therefore, VC is in NP.

We will now reduce a known NP-Complete problem to VC in polynomial time to show that VC is NP-Complete. Specifically, we will reduce the IS problem to VC problem. Given an instance of IS (G, K), it is transformed into an instance of VC $(G, n - K)$, where n is the number of vertices in G.

A subset S is an IS of G iff the set $V \setminus S$ is a vertex cover of G.

($\Rightarrow$) Let S be an IS of G. Consider any edge $(u, v) \in E$. Only one of the vertices u, v can be in S. Hence, at least one of u, v is in $V \setminus S$. Therefore, $V - S$ is a vertex cover.

($\Leftarrow$) Let $V \setminus S$ be a vertex cover of G. Consider a pair of vertices $u, v \in S$. There cannot be an edge between u and v because this edge would not be covered by $V \setminus S$. Therefore, S is an independent set of G.

3.6 Clique

A set of vertices U of G is a *clique*, if every pair of vertices in U has an edge in G. In the CLIQUE problem, given a graph G and an integer K, we have to decide if G has a clique of size K.

Claim: CLIQUE is NP-Complete

Proof: Given a set of vertices, we can check in polynomial time ($\binom{K}{2}$) if it is a clique or not. Therefore, CLIQUE problem is in NP.

Given an instance of IS (G, K), it is transformed into an instance of CLIQUE $(\bar{G}, K)$, where $\bar{G}$ is the complement graph of G. This transformation takes polynomial time (Looking at every pair of vertices and adding an edge or deleting an edge).

A subset S is an IS of G iff the set S is a clique in $\bar{G}$.

($\Rightarrow$) Let S be an IS in G. No two vertices in S share an edge in G. This implies that every pair of vertices in S share an edge in $\bar{G}$. Hence, S is a clique in $\bar{G}$

($\Leftarrow$) Let S be a clique in $\bar{G}$. Every pair of vertices in S has an edge in $\bar{G}$. Therefore, no two vertices in S share an edge in G. Hence, S is an IS in G.

3.7 Max-cut

A *cut* in a graph is defined as a partition of the vertices of the graph into two non-empty subsets. The *size* of the cut is defined as the number of edges crossing the cut.

The MAX-CUT problem is given a graph $G = (V, E)$ to find a cut with the maximum size.

Claim: This is NP-Hard

Proof: We show MAX-CUT is NP-Hard by reducing the problem of finding the minimum VC to the MAX-CUT problem.

Transformation: From the given graph G, a new graph G' is constructed as follows. A new vertex w is added to G. Every vertex u in G is connected by $deg(u) - 1$ parallel edges to w. Thus, the new graph G' has $|V| + 1$ vertices and $|E| + \sum_{u \in V}(deg(u) - 1)$ edges. This transformation takes polynomial time.

Let U and $V \setminus U + w$ be a cut in G'. We now prove that U and $V \setminus U + w$ is the max-cut in G' iff U is a minimum size vertex cover in G.

An edge in the graph is said to be incident on a set of vertices A if the edge has one of its vertices in A. The number of edges incident on U in G', $S_{G'}(U)$ is given by the following relation

$$S_{G'}(U) = S_G(U) + \sum_{u \in U}(deg(u) - 1), \tag{3.1}$$

where $S_G(U)$ is the number of edges incident on U with the vertex w removed. Simplifying the above equation, we obtain

$$S_{G'}(U) = S_G(U) + \sum_{u \in U} deg(u) - |U|. \tag{3.2}$$

Observe that the sum $S_G(U) + \sum_{u \in U} deg(u)$ is twice the number of edges with at least one vertex in U: (i) Say, only one vertex of an edge of G in U. Then, it is counted once in $S_G(U)$ and again in $\sum_{u \in U} deg(u)$. (ii) Say, both vertices of an edge of G in U. In this case, it contributes 2 to the sum $\sum_{u \in U} deg(u)$; one edge from each vertex to w. (iii) Those edges with none of the end points in U do not contribute to the sum. Therefore,

$$S_{G'}(U) = 2\{\text{number of edges of } G \text{ with at least one vertex in } U\} - |U|. \tag{3.3}$$

If U is a vertex cover of G, then

$$S_{G'}(U) = 2|E| - |U|. \tag{3.4}$$

($\Rightarrow$) From (3.4), if U is the minimum vertex cover, then $S_{G'}(U) \geq S_{G'}(T)$ for any other $T \subseteq V$.

($\Leftarrow$) Suppose, U and $V \setminus U + w$ is the max-cut in G'. We have to first prove that U is a vertex cover of G. It will then be the minumum from the relation in (3.4).

If U is not a vertex cover of G, then there is an edge (a, b) such that $a, b \in V \setminus U + w$. In this case, a new cut can be formed by adding one of the vertices of (a, b) to U. Say, we add the vertex b. This increases the size of the cut by $deg_G(b) - 1 + 1$. This also decreases the size by the number of edges incident on b of the form $(x, b), x \in U$. This can be at most $deg_G(b) - 1$ as there is at least one edge incident on b not in U; edge (a, b). There is a net increase in the size of the cut which contradicts the max-cut property of U and $V \setminus U + w$. Therefore, U is a vertex cover of G.