ICS 260—Fall, 2000–Class Notes 13: NP-complete problems

Notes Originally Developed by Sandy Irani
Modified by Michael B. Dillencourt

In this lecture, we will start with the fact that SAT is NP-complete, and prove a series of reductions, showing that a variety of other problems are also NP-complete.

1 3SAT is NP-complete

3SAT is the same as SAT, except with the restriction that every clause must have exactly three literals. We will show that this restriction does not change the complexity of the problem within a polynomial factor. That is, we will show that 3SAT is also NP-complete. Interestingly, when the restriction is made that every clause has exactly two literals (2SAT) the problem can be shown to be in P.

Theorem 1 *3SAT is NP-complete.*

Proof. The first part of an NP-completeness proof, is always to show that the problem is in NP. In this case, we could just 'guess' the truth assignment for the variables. If the set of clauses is satisfiable and we guess the correct satisfying assignment, we can check that every clause is satisfied in linear time.

Now we will show $SAT \propto 3SAT$. Every instance of SAT is characterized by a pair (U, C) denoting the set of variables and the set of clauses. We need to show a transformation f from instances of SAT to instances of 3SAT such that an input (U, C) to SAT is satisfiable if and only if $f(U, C)$ is satisfiable. Furthermore, $f(U, C)$ must be an instance of 3SAT. That is, it must be a set of variables and a set of clauses such that every clause has exactly three literals.

Let $C = \{c_1, \ldots, c_m\}$ be the set of clauses in an instance of SAT. For each clause c_i, we will produce a set of variables U_i and a set of clauses C_i. If $f(U, C) = (U', C')$, then $U' = U \cup (\cup_{i=1}^m U_i)$ and $C' = \cup_{i=1}^m C_i$.

The construction for U_i and C_i will depend on the number of literals k in c_i. Here are the cases:

$k = 1$: $c_i = \{z\}$. Then

$$U_i = \{y_{i,1}, y_{i,2}\}$$
$$C_i = \{\{z, y_{i,1}, y_{i,2}\}, \{z, y_{i,1}, \bar{y}_{i,2}\}, \{z, \bar{y}_{i,1}, y_{i,2}\}, \{z, \bar{y}_{i,1}, \bar{y}_{i,2}\}\}.$$

$k = 2$: $c_i = \{z_1, z_2\}$. Then

$$U_i = \{y_i\}$$
$$C_i = \{\{z_1, z_2, y_i\}, \{z_1, z_2, \bar{y}_i\}\}.$$

$k = 3$: $c_i = \{z_1, z_2, z_3\}$. Then

$U_i = \emptyset$

$C_i = \{\{z_1, z_2, z_3\}\}$.

$k > 3$: $c_i = \{z_1, z_2, \ldots, z_k\}$. Then

$U_i = \{y_{i,1}, y_{i,2}, \ldots, y_{i,k-3}\}$.

$C_i = \{\{z_1, z_2, y_{i,1}\}, \{\bar{y}_{i,1}, z_3, y_{i,2}\}, \{\bar{y}_{i,2}, z_4, y_{i,3}\}, \ldots, \{\bar{y}_{i,k-4}, z_{k-2}, y_{i,k-3}\}, \{\bar{y}_{i,k-3}, z_{k-1}, z_k\}\}$

Let $|c_i|$ denote the number of literals in clause c_i. The number of variables added for c_i is at most $|c_i| + 2$. The total number of clauses in C_i is at most $|c_i| + 3$. Thus, if there were n variables and m clauses in the original instance of SAT, there at most $n + 2m + \sum_{i=1}^{m} |c_i|$ variables and $3m + \sum_{i=1}^{m} |c_i|$ clauses in the new instance of 3SAT. This is polynomial (in fact linear) in the size of the original instance of SAT. Furthermore, computing the new instance takes time which is bounded by a polynomial in the size of the original instance of SAT.

Now we must establish that (U, C) is satisfiable if and only if (U', C') is satisfiable. Suppose (U, C) is satisfiable. Let t be a satisfying assignment such that every clause is satisfied. We will show how to set the variables in each U_i so that all the clauses in C_i will be satisfied. If there are 1, 2 or 3 literals in c_i, it doesn't matter how the extra variables will be set. The clauses in C_i will all be satisfied since each clause in C_i contains all the literals in c_i and at least one of these literals is true. If $|c_i| > 3$, let z_j be the literal in c_i which evaluates to true in the truth assignment. For $1 \leq l \leq j-2$, we will set $y_{i,l}$ to be true. For $j-1 \leq l \leq k-3$, we will set $y_{i,l}$ to be false. It can be easily verified that all the clauses in C_i will be satisfied.

Now suppose that (U', C') is satisfiable. Let t' be a truth assignment which satisfies all the clauses. Let t be the truth assignment which is just t' limited to the variables in U. We will prove that t satisfied (U, C). Consider an arbitrary clause $c_i = \{z_1, z_2, \ldots, z_k\}$ in C. If $k < 3$, then there must be at least one clause in C_i such that the literals corresponding to the y variables are all false. In this clause, one of the z literals must be true. This means that the clause c_i must be satisfied. If $k > 3$, if all the y_i variables are set to true, then it must be that either z_{k-1} or z_k is true, otherwise the last clause would not be satisfied. If $y_{i,1}$ is false, then either z_1 or z_2 must be true, or else the first clause would not be satisfied. Otherwise, let $y_{i,j}$ be the first variable which is false. The only way for the clause $\{\bar{y}_{i,j-1}, z_{j+1}, y_{i,j}\}$ to be satisfied is for z_{j+1} to be true. We have established that at least one literal in c_i must be true which means that c_i is satisfied.

■

2 Independent Set is NP-complete

An *independent set* in an undirected graph $G = (V, E)$ is a subset $I \subseteq V$ of the vertices such that no two vertices in I are connected by an edge. That is for $v, u \in I$, $\{u, v\} \notin E$. Consider the following problem, called the *Independent Set* problem.

Input: An undirected graph $G = (V, E)$ and an integer $B \geq 1$.

Question: Does G have an independent set of size at least B?

We shall prove that the independent set problem is NP-complete

The first task is to show that the independent set problem is in NP. To do this, we can just 'guess' the independent set. Then if the graph does have an independent set of size B and we have guessed correctly, we can verify in linear time that the number of nodes in the set we guessed is at least B and that no two vertices in the set are adjacent.

Now we must show that $3SAT \propto IND$. Given an arbitrary instance (U, C) of 3SAT, we shall produce an undirected graph $G = (V, E)$ and a bound B such that (U, C) is satisfiable if and only if there is an independent set in G of size at least B. Note that this must work for an arbitrary instance of 3SAT, but it may produce a graph G with a very special structure.

Let $\{c_1, c_2, \ldots, c_m\}$ denote the clauses in C. We will have $3m$ vertices in G, each corresponding to a literal in one of the clauses. Consider the clause $c_i = \{z_{i,1}, z_{i,2}, z_{i,3}\}$. We will have three vertices $v_{i,1}$, $v_{i,2}$ and $v_{i,3}$. We will also have edges $\{\{v_{i,1}, v_{i,2}\}, \{v_{i,2}, v_{i,3}\}, \{v_{i,3}, v_{i,1}\}\}$. So far we have m triangles in the graph. Then for any two nodes which correspond to opposite literals, we will add an edge between those two vertices. That is, if $z_{i,2} = \neg z_{j,3}$, then we will have an edge between $v_{i,2}$ and $v_{j,3}$. The bound B will just be m, the number of clauses in C.

The number of vertices in the graph is $3m$ and the number of edges is $O(m^2)$. Furthermore, the graph can be constructed in polynomial time.

Now we must show that (U, C) is satisfiable if and only if G has an independent set of size at least m. Let's start with a satisfying assignment for (U, C). Each clause c_i has at least one literal $z_{i,j}$ which is set to true. Pick one for each clause and add the corresponding vertex $v_{i,j}$ to the independent set. Since all m clauses are satisfied, we will have added m vertices to the independent set. We say that an edge is *violated* if we have chosen both vertices incident to the edge. Since we are choosing only one vertex per triangle, none of the edges in the triangles will be violated. Furthermore, since we started with a valid truth assignment, if there are two opposite literals, they can not both evaluate to true. This means we will not choose both of the corresponding variables. Thus, none of the additional edges will be violated.

Suppose we have an independent set of size m. It must be the case that there is exactly one chosen vertex from each triangle (i.e. clause) because we can not pick two vertices from any triangle. For each vertex chosen, we will set its corresponding literal to true. (If the literal is the negation of a variable, we will set the variable to false). If there are unassigned variables remaining, they are set arbitrarily. There are no contradictions in this assignment. That is, a literal can not be both set to true and false. This would correspond to setting two opposite literals to true which would correspond to having two vertices in the independent set which are adjacent. Furthermore, the set of clauses is satisfied because we have set to true, one literal from each clause.

3 3 Dimensional Matching and Knapsack

We introduce two more problems:

3 DIMENSIONAL MATCHING

Input: A set S of n items $\{s_1, \ldots, s_n\}$. A set F of m subsets of S. Each subset has three elements. $F = \{\{s_{i_{1,1}}, s_{i_{1,2}}, s_{i_{1,3}}\}, \ldots, \{s_{i_{m,1}}, s_{i_{m,2}}, s_{i_{m,3}}\}\}$.

Question: Is there a set of $n/3$ subsets $M \subseteq F$, such that the union of all the sets in M is S?

This problem is called three dimensional matching (3DM) or exact cover by 3 sets. The other problem we have seen before:

KNAPSACK

Input: A set S of n items. Each item s_i has a weight w_i and a value i. A bound W for the weight and a bound V for the value.

Question: Is there a subset $K \subseteq S$ such that the sum of the weights of the items in K is at most W and the sum of the values of the items in K is at least V.

We will omit the proof that 3DM is NP-complete. (The usual reduction for this problem is from 3SAT). Instead we will assume that 3DM is NP-complete and will prove that the knapsack problem is NP-complete.

Theorem 2 *The Knapsack problem is NP-complete.*

Proof. Knapsack is clearly in NP since we can just 'guess' the set of items we will choose. If there is a set of items that fits the constraints and we guess correctly, it is easy in polynomial time to sum up the weights of the chosen items and verify that their total weights is at most W. It is also easy to sum up the values of the items and verify that their total value is at least V.

Now we will show a transformation from an instance of 3DM to an instance of Knapsack. Let n be the number of items in the instance of 3DM. We can think of each 3-set as an n bit vector. Note that each vector has exactly 3 1's and the rest of the bits are 0. In this case, the bit-wise OR of two vectors corresponds to the union of the two sets. Thus, 3DM is asking are there $n/3$ 3-sets such that the bit-wise OR of their corresponding bit vectors is exactly the all 1's vector. Since we are only allowed $n/3$ vectors and there are only three bits per vector, we are asking if there are $n/3$ vectors such that for each bit place, exactly one of the vectors has a 1 in that place.

For the transformation, we would like an item for each 3-set in the 3DM problem. For each item, we will make $w_i = v_i$. Furthermore, the bounds for the weight and value are equal $V = W$. Thus, we are asking whether there is a subset of the items whose total value is exactly V. We will make the weight of each item the number corresponding to the bit vector representing the set. That is, each bit vector can be interpreted as a binary number. Now we want to know if there is a set of numbers which sum up to $2^n - 1$ which is the all 1's vector. The problem with the scheme as we have it is that in summing up a set of binary vectors, carry bits can overflow into other bits in the vector. To correct this problem, we will pad the binary representation of each number with

$\log m$ 0's between every bit. Since there are a total of m vectors, it is impossible for the carry sign to carry over into the bit representing the next element. Similarly, let V be the binary number obtained by repeating a 1 followed by $\log m$ 0's n times. Now if at most $n/3$ of the numbers sum to V it must be the case that their bit-wise OR is also V which means that there is a 3DM for the matching problem. Similarly, if there is a matching for the 3DM problem, the values of the corresponding items sum to V. ■

4 Maximum Cut

Recall that the problem of finding the minimum cut in a graph is in P since it is the same as finding the maximum flow for which we have seen polynomial time algorithms. We will now see that the maximization problem for finding cuts (finding the maximum cut) is NP-complete.

Here is the decision version of the problem:

MAXIMUM CUT

Input: An undirected graph $G = (V, E)$ and an integer k,

Question: Is there a partition of the vertices $(S, V - S)$ such that the number of edges crossing the cut $|\{\{x, y\} \mid x \in S, y \notin S\}|$ is at least k?

We will reduce from a slightly different form of 3SAT called Not-All-Equal SAT (NAESAT). Proving that NAESAT is NP-complete is a problem for the homework.

Not-All-Equal SAT

Input: A set U of variables and a set C of clauses with exactly three literals per clause.

Question: Is there a truth assignment such that in every clause there is at least one true and at least one false literal?

We will prove that max cut is NP-complete under the assumption that NAESAT is NP-complete.

Theorem 3 *MAX CUT is NP-complete.*

Proof. Given an instance of NAESAT (U, C), we will produce a graph G and a bound k such that there is n truth assignment in which every clause has at least one false and one true literal if and only if there is a cut in G with at least k edges.

In this graph, we will allow multiple edges between two vertices. If two vertices are separated by the cut, we will count all of the edges between the two vertices. If there are n variables in U, then we will have $2n$ vertices in G. Each variable u_i in G corresponds to a vertex u_i and a vertex $\bar{u}_i$. For each clause $\{z_1, z_2, z_3\}$, we will add three edges $\{\{z_1, z_2\}, \{z_1, z_3\}, \{z_3, z_2\}\}$. Note that the triangle of edges added for a clause will either contribute 2 or 0 edges to the cut. In addition, for each pair of opposite literals u_i and $\bar{u}_i$, we will add $2n_i$ edges between the vertices u_i and $\bar{u}_i$, where n_i is the total number of occurrences of either u_i or $\bar{u}_i$ in the clauses. The bound $k = 8m$.

Let's suppose that there is a cut $(S, V - S)$ such that the number of edges crossing the cut is at least k. Suppose that there is a pair of opposite literals u_i and $\bar{u}_i$ on the same side of the cut. They are contributing at most $2n_i$ edges to the cut. If we put u_i on the opposite side of the cut, the number of edges can not decrease (since we are adding at least $2n_i$ edges to the cut). In this manner separate each pair of opposite literals which are not already separated. The number of edges crossing the cut is still at least $8m$. There are $6m$ edges across the cut due to opposite literals (twice the number of literals in clauses). This means that there are $2m$ edges which cross the cut from the triangles. If we make a truth assignment where all the literals on one side of the cut are true and all the literals on the other side of the cut are false, we know that it is a valid truth assignment since all opposite literals are on opposite sides of the cut. Furthermore, each of the m triangles has two edges crossing the cut, indicating that there is at least one true and one false literal.

Now suppose that there is a truth assignment for (U, C) in which every clause has a true and a false literal. Let S be the set of vertices corresponding to literals which are true. We will have at least $8m$ edges crossing the cut, $6m$ from the edges between opposite literals and 2 for each triangle. ■