Lecture 14:
NP-complete versions of SAT

Valentine Kabanets

November 8, 2016

1 NP-complete versions of SAT

We showed last time that SAT is NP-complete, where SAT is to decide if a given propositional formula $\phi(x_1, \ldots, x_n)$ is satisfiable.

We will show that restricting the type of formulas to be CNFs or 3-CNFs (where each clause is of size 3 or of size at most 3) still leaves the problem NP-complete.

First, we consider the more general SAT version:

Circuit-SAT: Given a Boolean circuit $C(x_1, \ldots, x_n)$, decide if C is satisfiable.

Here a circuit has input gates $x_1, \ldots, x_n$, and other gates labeled by AND, OR, or NOT. The circuit has a single output gate.

We use the fact that SAT, and hence, Circuit-SAT, are NP-complete, to argue that CNF-SAT is also NP-complete, where

CNF-SAT: Given a CNF formula $\phi(x_1, \ldots, x_n)$, decide if ϕ is satisfiable.

Theorem 1. *CNF-SAT is NP-complete.*

Proof. Clearly, CNF-SAT is in NP. Thus it suffices to show that $Circuit - SAT \leq_p CNF - SAT$.

Let C be an arbitrary Boolean circuit with gates $g_1, \ldots, g_m$, where $g_1, \ldots, g_n$ are input gates and g_m is the output gate. For each g_j, introduce a Boolean variable y_j. For every $i > n$, define the Boolean formula $gate_i$ expressing that the value of y_i is equal to the value of the gate g_i. That is, if gate g_i is an AND gate with inputs g_{i_1} and g_{i_2}, then $gate_i$ is True iff $y_i \equiv y_{i_1} \wedge y_{i_2}$; similarly, for OR, and NOT gates.

Our final formula ϕ_C is defined as

$$\wedge_{i=n+1}^{m} gate_i \wedge \text{“}y_m \equiv 1\text{”}$$

It is left as an exercise to verify that C is satisfiable iff ϕ_C is satisfiable. Finally, it is easy to transform each formula $gate_i$ into a CNF formula on the corresponding 3 variables (or 2 variables in the case of the NOT-gate). So, our final formula is a CNF formula. □

Consider **3SAT:** Given a 3-cnf formula $\phi(x_1, \ldots, x_n)$, decide if ϕ is satisfiable.

Here, a 3-cnf is a cnf where each clause is of size at most 3 (contains at most 3 literals).

If we look inside the earlier reduction $Circuit - SAT \leq_p CNF - SAT$, we will see that the CNF formula produced there is in fact a 3-cnf! Thus, we get the following

Corollary 1. *3-SAT is NP-complete.*

Remark 1. *Sometimes, by 3-SAT, people mean the satisfiability question for 3-cnfs where each clause is of size* exactly *3 (rather than at most 3). We leave it as an exercise to show that this version of 3-SAT is also NP-complete!*

Consider **NAE-3SAT:** Given a 3-cnf formula, decide if there is a satisfying assignment such that each clause contains at least one false literal (and at least one true literal).

Here NAE-SAT stands for "Not All Equal" SAT, meaning that an assignment exists under which no clause of the formula has all equal literals.

Theorem 2. *NAE-SAT is* NP*-complete.*

Proof. As usual, NAE-SAT is in NP (easy). To show that NAE-SAT is NP-hard, we will slightly modify our previous reduction from Circuit-SAT to SAT.

Recall that this reduction associated a variable y_i with each gate of a given circuit, and produced a formula for each gate as follows. If i is an AND gate with inputs j and k, then we create the formula expressing that "$y_i \equiv y_j \wedge y_k$". The last formula can be written as the CNF

$$(\bar{y_i} \vee y_j) \wedge (\bar{y_i} \vee y_k) \wedge (\bar{y_k} \vee \bar{y_j} \vee y_i) \tag{1}$$

Now, our modified formula will be the formula produced by the "Circuit-SAT$\leq$SAT" reduction where each clause of size 1 or 2 gets a new literal z added to it (the same for all clauses). We claim that the new formula is satisfiable in the NAE-SAT sense iff the original formula (without the z) is satisfiable.

Suppose the modified formula is satisfied by assignment a in the NAE-SAT sense. Then $\bar{a}$ is also a satisfying assignment for this formula. Let's pick the one of these two satisfying assignments that makes z False. This assignment will also satisfy the original formula (without the z) (Check this!)

For the other direction, starting with a satisfying assignment for the original formula, we create a satisfying assignment for the modified formula by setting z to False. We need to argue that this is a satisfying assignment in the NAE-SAT sense, i.e., that every clause has at least one false literal. Here we use the fact that the original formula has a very particular form. It has groups of clauses associated with every gate of the circuit from which this formula was constructed (by the "Circuit-SAT to SAT" reduction). For example, an AND gate will be associated with three clauses of the type given above in formula (1). The two clauses of size 2 in formula (1) will have z added to them in the modified formula, and since z is set to False, they both will have a false literal. The remaining clause of size 3 must also have a false literal. Suppose it does not. Then it is easy to see that both size-2 clause would need to be false, which contradicts the fact that we started with a satisfying assignment. The case of an OR gate, and a NOT gate can be argued similarly. □

Remark 2. *Above, NAE-3SAT talks about 3-cnfs where each clause is of size at most 3. It is possible to show that NAE-3SAT for 3-cnfs where each clause is of size exactly 3 is also NP-complete. (Exercise!)*

2 NP-completeness of 3-COL

3-COL $= \{G \mid G$ is a 3-colorable graph$\}$ (recall that a 3-colorable graph is a graph whose vertices may be colored with colors 0,1, and 2 in such a way that the endpoints of every edge receive different colors).

Theorem 3. *3-COL is* NP*-complete.*

Proof. We need to prove that

1. 3-COL is in NP, and

2. 3-COL is NP-hard (i.e., every language $L \in NP$ reduces to 3-COL).

We prove (1) by giving the following NP algorithm for 3-COL: Given a graph G, nondeterministically guess an assignment of colors 0,1,2 to the vertices of G; check (in deterministic polytime) that the guessed coloring is proper, i.e., that no edge has both of its endpoints colored with the same color.

To prove (2), we reduce NAE-3SAT to 3-COL. Given a 3-CNF formula $\phi(x_1, \dots, x_n)$, we construct a graph G_ϕ such that $\phi \in$NAE-SAT iff $G_\phi \in$3-COL. Our graph G_ϕ will have
vertices:

- $x_1, \bar{x}_1, x_2, \bar{x}_2, \dots, x_n, \bar{x}_n$ (i.e., one vertex for each literal),

- a vertex u, and

- a triple of vertices for each 3-clause $C_j = (l_{j_1} \vee l_{j_2} \vee l_{j_3})$ labeled by $v(j, l_{j_1}), v(j, l_{j_2}), v(j, l_{j_3})$, respectively.

Our graph will have
edges:

- $(x_i, \bar{x}_i)$ for each $1 \le i \le n$,

- (u, x_i) and $(u, \bar{x}_i)$ for each i,

- the triple of vertices corresponding to a clause will be connected to each other (i.e., every clause $C_j = (l_{j_1} \vee l_{j_2} \vee l_{j_3})$ corresponds to a triangle on the vertices $v(j, l_{j_1}), v(j, l_{j_2}), v(j, l_{j_3})$, and

- for every clause $C_j = (l_{j_1} \vee l_{j_2} \vee l_{j_3})$, there are three edges $(v(j, l_{j_1}), l_{j_1})$, $(v(j, l_{j_2}), l_{j_2})$, and $(v(j, l_{j_3}), l_{j_3})$ (i.e., each vertex in a clause-triangle is connected to the corresponding literal-vertex).

We now prove the correctness of our reduction. First, assume that G_ϕ is 3-colorable. Without loss of generality, the vertex u is colored with color 2. So, each of the literal-vertices connected to u will get colors 0 or 1. Our truth assignment will set variable x_i to True, if vertex x_i is colored with color 1; and to False, if vertex x_i is colored with 0. Now we argue that this assignment is satisfying for ϕ in the NAE-SAT sense, i.e., that every clause of ϕ has at least one true literal and at least one false literal.

Consider any clause $C_j = (l_{j_1} \vee l_{j_2} \vee l_{j_3})$ corresponding to the triangle on vertices

$$v(j, l_{j_1}), v(j, l_{j_2}), v(j, l_{j_3})$$

of G_ϕ. Suppose that all literals in C_j are assigned True by our truth assignment. Then it means that the vertices $l_{j_1}, l_{j_2}, l_{j_3}$ are all colored with color 1. So color 1 cannot be used to color the vertices of the triangle on $v(j, l_{j_1}), v(j, l_{j_2}), v(j, l_{j_3})$. But we cannot color a triangle with just two colors! A contradiction. So, at least one literal in clause C_j is assigned False. A similar argument shows that at least one literal in C_j is assigned True. So ϕ is satisfied in the NAE-SAT sense.

Now we prove the other direction. Given an assignment a to ϕ which satisfies ϕ in the NAE-SAT sense, we define a coloring for G_ϕ as follows. The vertex u gets color 2. A vertex x_i gets color 1, if x_i is set to True by the assignment a, and color 0 otherwise. Consider the triangle corresponding to each clause $C_j = (l_{j_1} \vee l_{j_2} \vee l_{j_3})$. Since assignment a is satisfying in the NAE-SAT sense, there is at least one true literal and at least one false literal in C_j. W.l.o.g., assume that l_{j_1} is True, and l_{j_2} is False. Then we color the vertex $v(j, l_{j_1})$ with color 0, the vertex $v(j, l_{j_2})$ with color 1, and the vertex $v(j, l_{j_3})$ with color 2. It is not hard to verify that this coloring is indeed a proper 3-coloring of our graph. $\square$

3 "3 vs. 2"

We saw that 3-SAT and 3-COL are NP-complete. What about 2-SAT and 2-COL? Here, 2-SAT is naturally defined as: given a 2-cnf formula (where each clause is of size at most 2), decide if it's satisfiable. Similarly, 2-COL is the problem to decide if a given graph is 2-colorable (equivalently, is bipartite). You must have seen in your Algorithms course that 2-COL has an efficient algorithm, and so 2-COL is in P. It is left as an exercise for you to show that 2-SAT is also in P.

Thus, we have a sharp dividing line between easy (polytime) problems and hard (NP-complete) problems, where a small change in the formulation of the problem would cross the line.

On the other hand, for any $k \geq 3$, the problems k-SAT and k-COL remain NP-complete. (Exercise!)