One in Three SAT (1in3SAT) Problem

Zhiyuan Zhan
School of Electrical Engineering and Computer Science
Washington State University
zzhan@eeecs.wsu.edu

1 1in3SAT Problem Definition

Given: A collection of clauses, $c_1, c_2, \ldots, c_m$, each is a disjunction of at most three literals.

Question: Is there a through assignment, which set exactly one literal to true in each clause, that makes the conjunction of those clauses, $c_1 \land c_2 \land \ldots \land c_m$, true?

2 Claim and Proof

Claim: This problem, 1in3SAT, is NP-complete.

Proof:
Assumption: 3SAT problem is NP-complete.

1. Let’s first prove $1in3SAT \in NP$.

   We can easily prove this by showing that there is an algorithm $A$, which will run in polynomial time of the input (a given assignment) and output whether this assignment satisfies the requirements of the question or not.

   Let’s simply call this kind of algorithm the grading algorithm (GA).

   Since 3SAT is NP-complete, 3SAT $\in$ NP. Suppose $A$ is the GA for 3SAT, we add one additional check per clause to $A$ in order to verify that exactly one literal is set to true. Because $A$ runs in polynomial time of the input assignment, this “new” $A$ will run in polynomial time, too. So this “new” $A$ will serve as the GA for 1in3SAT problem because 1in3SAT has the same description as 3SAT, except that a satisfying truth assignment must set exactly one literal to true in each clause, which will be checked in the “new” $A$.

2. Let’s then prove $1in3SAT$ is NP-hard.

   We prove this by reducing 3SAT into 1in3SAT.

   The transformation is: for each of the $c_i = \{x_{i1}, x_{i2}, x_{i3}\}$, we add 4 new variables, $a_i, b_i, c_i, d_i$ and produce three new clauses:

   \[
   \{x_{i1}, a_i, b_i\}, \{x_{i2}, b_i, c_i\}, \{x_{i3}, c_i, d_i\}
   \]

   Suppose the original $c_1 \land c_2 \land \ldots \land c_m$ has $n$ variables and $k$ clauses, our transformation will produce an instance of $n+4k$ variables and $3k$ clauses. This transformation is obviously carried out in polynomial time.
Then, we need to prove that our transformed instance will have a solution (truth assignment) under $lin3SAT$ condition iff the original instance does under $3SAT$ condition.

($\implies$) Firstly, suppose our transformed instance has a solution, which means exactly one literal per clause is set to true, we want to show that the original instance will have a solution, too.
Suppose in the original instance clause $c_i$, $x_{i1}$, $x_{i2}$, $x_{i3}$ are all set to false. This means that in the second clause we produce, either $b_i$ or $c_i$ (but not both) must be true, otherwise it's a contradiction to exactly one literal has to be true. So either in the first or the third transformed clause, there will be at least two variables set to true (either $x_{i1}$ and $b_i$ or $x_{i3}$ and $c_i$, but not both). This is a contradiction to our $lin3SAT$ condition. So in the original clause $c_i$, at least one of three literals has to be true. Therefore the original instance must have a solution under $3SAT$ condition.

($\impliedby$) Secondly, suppose the original instance $c_1 \land c_2 \land \ldots \land c_m$ has a solution, there must be at least one literal set to be true in each clause.

Suppose in clause $c_i$,
(a) $x_{i2}$ is set to true, we can set $b_i$ and $c_i$ to false in the second transformed clause and set $a_i = x_{i1}$ and $d_i = x_{i3}$.
(b) $x_{i2}$ is set to false and both $x_{i1}$ and $x_{i3}$ are set to true, we can set $a_i$ to true, $b_i$ to false, $c_i$ to true and $d_i$ to false.
(c) only $x_{i1}$ is set to true, we can set $b_i$ to true, and $a_i$, $c_i$ and $d_i$ false.
(d) only $x_{i3}$ is set to true, we can set $d_i$ to true, and $a_i$, $b_i$ and $c_i$ false.

In all these cases, the three transformed clauses corresponding to $c_i$ will have exactly one literal set to true in each clause. Therefore, the transformed instance will have a solution under $lin3SAT$ condition.

Now we have that $lin3SAT$ is in $NP$, and $3SAT$ could be reduced to $lin3SAT$ via a mapping which clearly costs polynomial time. Hence $lin3SAT$ is $NP$-complete.

*End of Proof*