NP-Complete Problems 1

Eryk Zagorski

CS 501 - University of Illinois at Chicago

January 23, 2020

Intro

- We know we can prove a problem is NP-Complete through reduction
- What can we do if we don't have a problem to reduce to?
- We need a problem that can be used to solve every other NP problem
- To do this we can use Boolean SAT

What is SAT/Boolean SAT?

SAT is short for SATisfiability.
Given a set of (boolean) variables,

$$v_1, v_2 ... v_n$$

and a boolean statement,

$$S = v_1 \wedge (v_2 \vee v_3) ... (v_3 \wedge v_{n-1})$$

Does there exist an assignment to $v_1, v_2 ... v_n$ s.t. S is true?

What is SAT/Boolean SAT?

For example, if we have

$$v_1, v_2, v_3$$

and

$$S = \neg v_1 \wedge (v_2 \vee v_3)$$

S would be true if

$$v_1 = 0, v_2 = 1, v_3 = 0$$

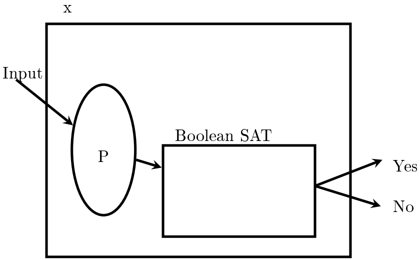
so, S is satisfiable

Proving Boolean SAT is NP Complete

We must prove that for every problem in NP, each problem can be solved using SAT. Formally:

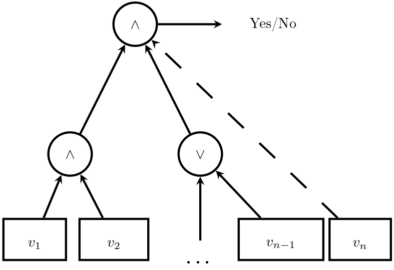
$$\forall x \in NP$$

$$x \leq_p SAT$$



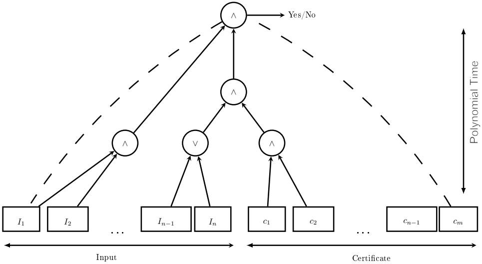
Proving Boolean SAT is NP Complete

To do this, first we will describe the verification algorithm for any Boolean SAT problem as a "compiled" polynomial time algorithm as can be seen in the diagram on the right.
This is polynomial because as we go up in the tree, for every two levels (in case of a a "not") we are guaranteed to reduce the number of inputs.



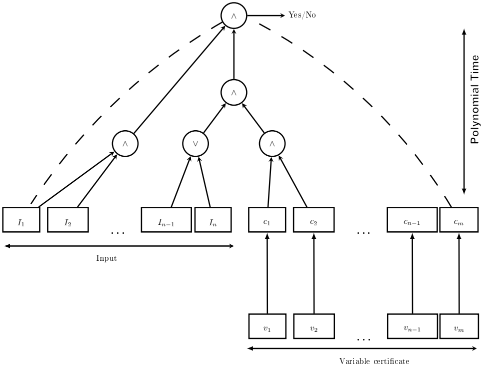
Proving Boolean SAT is NP Complete

Now we can use the same exact idea to verify any other NP problem by feeding its inputs and a certificate into our Boolean SAT "verifier".
This allows us to verify any NP problem using Boolean SAT.



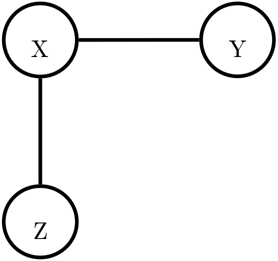
Proving Boolean SAT is NP Complete

Now by simply varying our certificate, we can try different values until we solve the decision version of any NP problem.



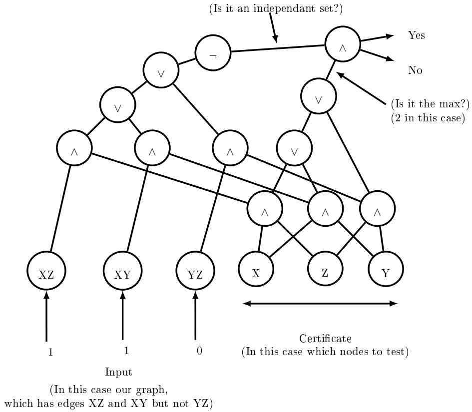
Brief Example: Using Boolean SAT to Solve MIS

Lets prove $MIS \leq_p SAT$
On the right is the graph we would like to solve using SAT
In this graph we have 3 possible edges: xy, xz, and yz. We will use these as boolean inputs into our SAT algorithm.
Our certificate will be which nodes we want to check are a maximum independent set.



Brief Example: Using Boolean SAT to Solve MIS

Now we can create a general SAT problem that takes our graph as an input, a selection of nodes as a certificate and outputs whether or not the certificate is the maximum independent set.
As mentioned earlier, we can then just vary the certificate until the MIS is found.



Proving MIS is NP-Complete

Now that we have an NP complete problem, we can start expanding our set of NP complete problems by reducing other problems to SAT. We know that 3-SAT can be reduced to SAT, so we won't show the proof for that. But we will use 3-SAT to prove MIS is NP Complete. So as always we must first show that:

$$MIS \in NP$$

which is trivial as we can just check that no nodes are connected and then we must show that that:

$$3SAT \leq_p MIS$$

What is 3-SAT?

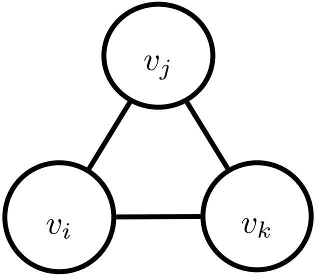
3-SAT is much like SAT but has a standard format for variables where 3 variables are OR'd into a clause and all clauses are AND'd, formally:
Given variables: $v_1, v_2 ... v_n$
and clauses: $c_1, c_2 ... c_m$
where clauses are defined as: $c_i = v_i \vee v_j \vee v_k$
we have a statement: $S = c_1 \wedge c_2 \wedge ... \wedge c_m$
is there an assignment of variables which makes S true?
e.g if we have a statement:

$$S = (v_1 \vee v_2 \vee \neg v_3) \wedge (v_2 \vee \neg v_1 \vee v_3) \wedge (\neg v_1 \vee \neg v_2 \vee \neg v_3)$$

$v_1 = 1, v_2 = 1, v_3 = 0$ would make the S true. Generally, we must have one true variable from each clause to satisfy the statement.

Reducing 3-SAT to MIS

First, for each clause from our 3-SAT statement we will create a gadget. So, $\forall c_i = v_i \vee v_j \vee v_k$ we create:



Note, for these gadgets, the maximum independent set is any single node, which means when we run MIS, only one node from our gadget can be selected.

Reducing 3-SAT to MIS

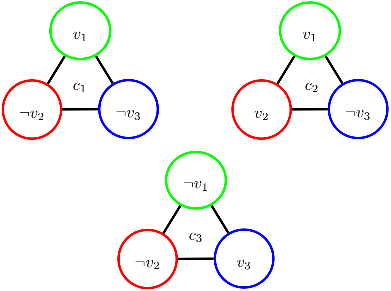
Now lets create an example with 3 clauses to show how our reduction will work:

$$c_1 = v_1 \vee \neg v_2 \vee \neg v_3$$

$$c_2 = v_1 \vee v_2 \vee \neg v_3$$

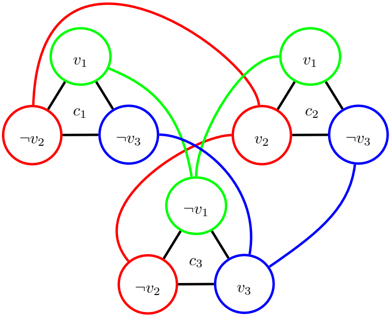
$$c_3 = \neg v_1 \vee \neg v_2 \vee v_3$$

And we'll create the three gadgets for our clauses:



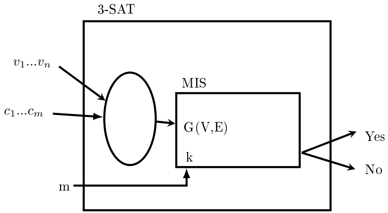
Reducing 3-SAT to MIS

Now all we have to do is connect all variables to their inverse.
For 3-SAT to return true we need two things to be true: there must be at least one true variable in each clause and each variable must be either true or false.
By creating the gadgets and setting k to m (number of clauses), MIS must choose one node from each gadget, and by connecting the variables to their inverse, MIS will only be able to select true or false for each variable.

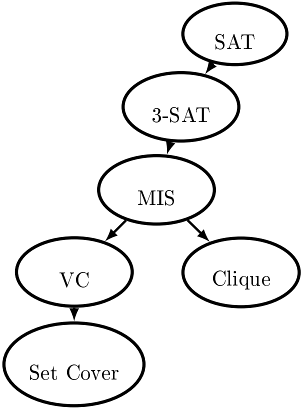


Reducing 3-SAT to MIS

So our reduction looks like this:

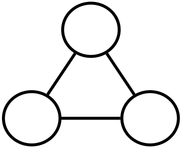


And our tree of NP complete problems grow. We will briefly cover Clique next.

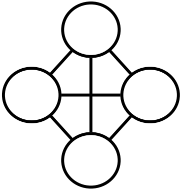


Clique

Given a graph G(V,E), find the largest clique where a clique of size k is a complete graph of k nodes.

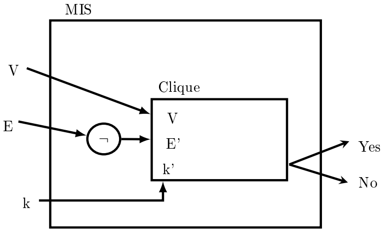


A clique of size 3



A clique of size 4

We can reduce MIS to clique with minimal effort by simply taking the complement of the edges:



The complement is:

$\forall (u,v) \notin E$ add $(u,v) \in E'$