Proving $\mathcal{NP}$-Completeness

We show problems are $\mathcal{NP}$-complete by reducing from known $\mathcal{NP}$-complete problems.

Proving $\mathcal{NP}$-Completeness by Reduction

To prove a problem is $\mathcal{NP}$-complete, use the earlier observation:

If S is $\mathcal{NP}$-complete, $T \in \mathcal{NP}$ and $S \leq_P T$, then T is $\mathcal{NP}$-complete.

Proof that `3SAT` *is* $\mathcal{NP}$*-complete*

Recall `3SAT`:

Input: ϕ a boolean formula in 3CNF
Question: is there a satisfying assignment?

The language `3SAT` is a ***restriction*** of `SAT`, and so `3SAT` $\in \mathcal{NP}$.

Reducing 3SAT *to* SAT

We reduce SAT to 3SAT. The task is to describe a polynomial-time algorithm for:

input: a boolean formula ϕ in CNF
output: a boolean formula ψ_ϕ in 3CNF such that ϕ is satisfiable exactly when ψ_ϕ is.

Substituting Clauses

We replace each clause C of ϕ by family D_C of clauses that preserves satisfiability.

For example, say $C = a \vee b \vee c \vee d \vee e$. One can simulate this by

$$D_C = (a \vee b \vee x) \wedge (\bar{x} \vee c \vee y) \wedge (\bar{y} \vee d \vee e)$$

where x and y are new variables. Need to verify:

1) If C is FALSE, then D_C is FALSE; and
2) If C is TRUE, then one can make D_C TRUE.

Clauses of other sizes are handled similarly.

The Conclusion

So this yields ψ_ϕ in 3CNF. If ϕ is satisfiable, then there is assignment where each clause C is TRUE; this can be extended to make each D_C TRUE. Further, if assignment evaluates ϕ to FALSE, then some clause say C' must be FALSE and thus the corresponding family $D_{C'}$ in ψ_ϕ is FALSE.

The last thing to check is that the conversion process can be encoded as a polynomial-time algorithm. Thus, we have shown that SAT reduces to 3SAT, and so 3SAT is $\mathcal{NP}$-complete.

Proof that DOMINATION *is* $\mathcal{NP}$*-complete*

Recall that a dominating set D is such that every other node is adjacent to a node in D; and that the DOMINATION problem is:

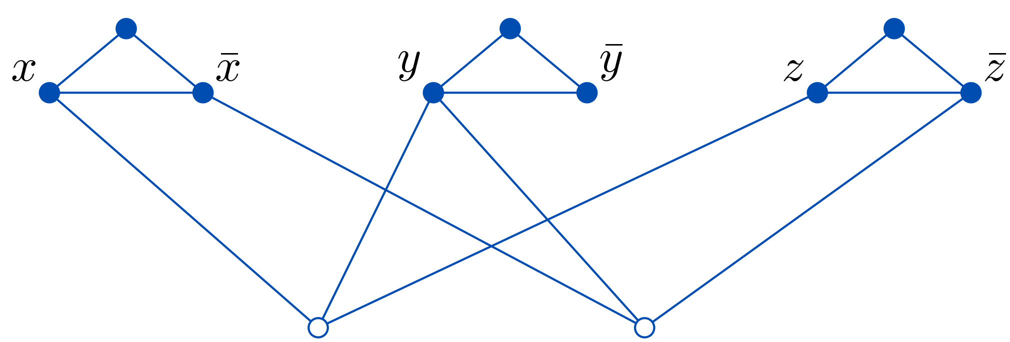
Input: graph G and integer k
Question: is there dominating set of at most k nodes?

We reduce 3SAT to DOMINATION. That is, we describe a procedure that takes boolean formula ϕ, and produces graph G_ϕ and integer k_ϕ such that ϕ is satisfiable exactly when there is a dominating set of G_ϕ of k_ϕ nodes.

Construction of Graph

Suppose ϕ in 3CNF has c clauses and m variables. For each clause, create a node. For each variable v, create a triangle with one node labeled v and one labeled $\bar{v}$. Then for each clause, join the clause-node to the three nodes corresponding to its literals. The result is graph G_ϕ.

For example, the graph for $(x \vee y \vee z) \wedge (\bar{x} \vee y \vee \bar{z})$:



The Reduction

Set $k_\phi = m$ (the number of vars). Claim: the mapping ϕ to $\langle G_\phi, k_\phi \rangle$ is the desired reduction.

If ϕ has satisfying assignment, then let D be the m nodes corresponding to TRUE literals in the assignment. Then each triangle is dominated, as is each clause-node. So D is dominating set.

Conversely, suppose G_ϕ has dominating set D of size m. Then D must be one node from each triangle, and every clause must be connected to one literal in D. So setting all the literals corresponding to nodes in D to TRUE is satisfying.

The Conclusion

That is, we have shown that 3SAT reduces to DOMINATION, and so DOMINATION is $\mathcal{NP}$-complete.

Gadgets

The above reduction illustrates a common pattern. To reduce from `3SAT`, create a "*gadget*" for each variable and a "*gadget*" for each clause, and connect them up somehow.

Proof that `SUBSET_SUM` *is* $\mathcal{NP}$*-complete*

Recall that input to Subset sum problem is set $A = \{a_1, a_2, \ldots, a_m\}$ of integers and target t. The question is whether there is $A' \subseteq A$ such that elements in A' sum to t.

We prove this problem is $\mathcal{NP}$-complete. This is again a reduction from `3SAT`. The previous example suggests the approach: define numbers x_i and $\bar{x}_i$ and a target t such that one can take only one of x_i and $\bar{x}_i$, and then some constraint is to be satisfied.

Vector Construction

Suppose that one has ***vectors*** instead of numbers. Two vectors (of same length) can be added component-wise. The question now is whether there is subset whose sum equals a specified vector.

Suppose input ϕ has c clauses and m variables. The vectors will have length $c + m$. For each vector, the first m positions will specify which variable by a 1 in the appropriate position. The second part records the clauses each literal is in.

Example Vectors

For example, if ϕ is

$$(x_2 \vee x_3 \vee \bar{x}_4) \wedge (x_1 \vee \bar{x}_3 \vee x_4) \wedge (x_1 \vee \bar{x}_2 \vee x_4)$$

then vectors corresponding to the variables are

$x_1 = (1,0,0,0;0,1,1)$ and $\bar{x}_1 = (1,0,0,0;0,0,0)$
$x_2 = (0,1,0,0;1,0,0)$ and $\bar{x}_2 = (0,1,0,0;0,0,1)$
$x_3 = (0,0,1,0;1,0,0)$ and $\bar{x}_3 = (0,0,1,0;0,1,0)$
$x_4 = (0,0,0,1;0,1,1)$ and $\bar{x}_4 = (0,0,0,1;1,0,0)$

Constructing the Target

A target of all 1's would force selection of exactly one of each variable and its negation. However, some clauses might have multiple true literals. So define t as all 1's for variables and all 3's for clauses: $t = (1, 1, 1, 1; 3, 3, 3)$.

Then add ***slack variables***. These are vectors that one can use to round sum up to t. Specifically, add *two* copies of each clause:

$c_1 = (0, 0, 0, 0; 1, 0, 0)$ and $c_1' = (0, 0, 0, 0; 1, 0, 0)$, etc.

Note that to reach 3 in a component, at least one 1 must be supplied by a literal.

The Reduction for Vectors

That is, we have built a set of vectors and a target vector such that there is a subset of vectors that sums to the target vector exactly when the boolean formula has a satisfying assignment.

(Well, actually we do have to argue this both ways.)

Conversion to Numbers

Finally, we go from vectors to numbers. Just think of the vector as the number in decimal:

$t = 1111333$ and $x_1 = 1000011$, $\bar{x}_1 = 1000000$, $x_2 = 0100100$, $\bar{x}_2 = 0100001$, etc.

A worry is that one might be able to reach the target in unintended way, but that does not happen. So we have shown a reduction from `3SAT` to `SUBSET_SUM`, and so `SUBSET_SUM` is $\mathcal{NP}$-complete.

Believe it or not, these reductions become routine, eventually.

Practice

Show that VERTEX_COVER is $\mathcal{NP}$-complete.

(Recall that the removal of a vertex cover destroys every edge, and that the input to VERTEX_COVER is graph G and integer k.)

(Hint: Reduce from 3SAT using two connected nodes for each variable and three connected nodes for each clause.)

Solution to Practice

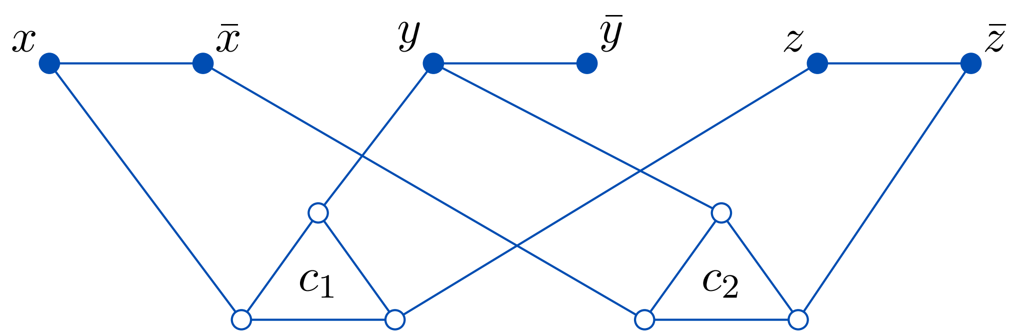
We first show that `VERTEX_COVER` is in $\mathcal{NP}$. The nondeterministic program guesses k nodes and then checks they form a vertex cover.

We then reduce `3SAT` to `VERTEX_COVER`. We describe a procedure to take a boolean formula ϕ, and produce graph G_ϕ and integer k_ϕ, such that ϕ is satisfiable exactly when there is vertex cover of G_ϕ of k_ϕ nodes.

Solution: Constructing the Graph

Assume ϕ has c clauses and m variables. For each variable v, create two adjacent nodes labeled v and $\bar{v}$. For each clause, create three adjacent nodes and join each to a literal in the clause.

For example, the graph G_ϕ for $(x \vee y \vee z) \wedge (\bar{x} \vee y \vee \bar{z})$:



Solution: The Reduction

Let $k_\phi = m + 2c$. Claim: the mapping ϕ to $\langle G_\phi, k_\phi \rangle$ is the desired reduction. The main part is to show that the mapping preserves the answer.

Suppose G_ϕ has vertex cover D of size k_ϕ. It contains at least one node from each node-pair and two nodes from each clause-triangle. Since D has size $m+2c$, this is exactly what D is. Thus when we remove D, for each clause one node remains, and so the other end of that edge is in D. That is, the literals in D are a satisfying assignment.

Solution Continued

Conversely, suppose ϕ has a satisfying assignment. Then let D be the m nodes corresponding to the TRUE literals in the assignment. Then each clause-triangle is dominated. So one can add two nodes from each clause-triangle and all edges incident with that clause are taken care of. It follows that G_ϕ has a vertex cover of size $m + 2c$.

That is, we have shown that `3SAT` reduces to `VERTEX_COVER`, and so `VERTEX_COVER` is $\mathcal{NP}$-complete.

Summary

A new problem can be proven $\mathcal{NP}$-complete by reduction from a problem already known to be $\mathcal{NP}$-complete.