Examples of $\mathcal{NP}$-Complete Problems

AC32008: Theory of Computation

Vladimir Janjic

University of Dundee
QMB 1.22
vjanjic001@dundee.ac.uk

Overview

- 3SAT is $\mathcal{NP}$-Complete
- VERTEX-COVER is $\mathcal{NP}$-Complete

$\mathcal{NP}$-Completeness

- $\mathcal{NP}$-Complete problems have a feature that every $\mathcal{NP}$ problem is polynomial-time reducible to them.
- We have seen what impact the existence of $\mathcal{NP}$-Complete problems has on the $\mathcal{P} = \mathcal{NP}$ problem.
- We have also discussed practical impact of knowing that some problem is $\mathcal{NP}$-Complete.
- We have proven that SAT is $\mathcal{NP}$-Complete problem.

How to Prove that a Problem is $\mathcal{NP}$-Complete

To prove that some problem/language L is $\mathcal{NP}$-Complete we need to show that

1. $L \in \mathcal{NP}$

and either

2. For some language A that is $\mathcal{NP}$-Complete, $A \propto L$.

or

3. For every language $A \in \mathcal{NP}$, $A \propto L$.

3SAT Problem

- 3SAT problem is similar to SAT problem, except that the Boolean formula has to be in a specific format.
- A literal is a Boolean variable or a negated boolean variable (as in x or $\overline{x}$).
- A clause comprises of literals connected with $\vee$'s, e.g. $(x_1 \vee \overline{x_2} \vee \overline{x_3} \vee x_4)$.
- A Boolean formula is in conjuctive normal form, also called a cnf-formula, if it comprises clauses connected by $\wedge$'s, e.g.

$$(x_1 \vee \overline{x_2} \vee \overline{x_3} \vee x_4) \wedge (x_5 \vee x_6) \wedge (\overline{x_1} \vee x_5 \vee \overline{x_6}).$$

- A cnf-formula is a 3cnf-formula if all the clauses have exactly 3 literals, e.g.

$$(x_1 \vee \overline{x_2} \vee x_3) \wedge (\overline{x_1} \vee x_4 \vee x_5) \wedge (x_2 \vee \overline{x_3} \vee \overline{x_5}).$$

3SAT Problem

3SAT problem is to test whether a 3cnf-formula is satisfiable. Formally, language 3SAT is defined as

$$\text{3SAT} = \{\langle\phi\rangle | \phi \text{ is a satisfiable 3cnf-formula}\}.$$

3SAT is $\mathcal{NP}$-Complete

Theorem

3SAT is $\mathcal{NP}$-Complete.

Proof

- We need to prove that 3SAT $\in \mathcal{NP}$ and that for some problem A that is $\mathcal{NP}$-Complete, $A \propto$ 3SAT.
- To prove that 3SAT is in $\mathcal{NP}$, we observe that SAT is in $\mathcal{NP}$ and instances of 3SAT are just special instances of SAT.
- Therefore, if there is polynomial-time NDTM that solves SAT, the same NDTM can be used to solve 3SAT too.
- Because of this, 3SAT is in $\mathcal{NP}$.

3SAT is $\mathcal{NP}$-Complete

Theorem

3SAT is $\mathcal{NP}$-Complete.

Proof

- We will prove that that SAT $\propto$ 3SAT
- We have to prove that every instance ϕ_1 of SAT is reducible in polynomial-time (with respect to the size of ϕ_1) to an instance ϕ_2 of 3SAT, such that ϕ_1 is satisfiable if and only if ϕ_2 is satisfiable.
- An instance ϕ_1 of SAT is a formula where we have some combination of $\wedge, \vee$ and $\neg$ operators on variables.
- We need to transform (in polynomial time with respect to the number of variables) this formula into a formula ϕ_2 that is a 3cnf-formula.
- Using the $(a \wedge b) \vee c = (a \vee c) \wedge (b \vee c)$ identity, we can easily transform any formula (in polynomial time) into a formula of the type $c_1 \wedge c_2 \wedge c_3 \wedge \cdots \wedge c_n$, where c_i are clauses.

Let

$$\phi_1 = ((x_1 \wedge x_2) \vee (\overline{x_2} \vee x_3 \vee x_4)) \wedge (x_3 \vee \overline{x_1}).$$

Transform ϕ_1 into 3cnf-formula ϕ_2 such that ϕ_1 is satisfiable if and only if ϕ_2 is satisfiable

Remembering that $(a \wedge b) \vee c = (a \vee c) \wedge (b \vee c)$, we have

$$\begin{aligned}\phi_1 &= ((x_1 \wedge x_2) \vee (\overline{x_2} \vee x_3 \vee x_4)) \wedge (x_3 \vee \overline{x_1}) \\ &= (x_1 \vee \overline{x_2} \vee x_3 \vee x_4) \wedge (x_2 \vee \overline{x_2} \vee x_3 \vee x_4) \wedge (x_3 \vee \overline{x_1}) \\ &= (x_1 \vee \overline{x_2} \vee x_3 \vee x_4) \wedge (x_3 \vee \overline{x_1})\end{aligned}$$

Theorem

3SAT is $\mathcal{NP}$-Complete.

Proof

- Using the $(a \wedge b) \vee c = (a \vee c) \wedge (b \vee c)$ identity, we can easily transform any formula (in polynomial time) into a formula of the type $c_1 \wedge c_2 \wedge c_3 \wedge \cdots \wedge c_n$, where c_i are clauses.
- However, c_i does not necessarily have exactly 3 literals.
- Let $c_i = (a_1 \vee a_2 \vee a_3 \vee \cdots \vee a_k)$, where $k > 3$.
- Let us introduce new variables $b_1, b_2, \ldots, b_{k-3}$ and let us consider

$$\begin{aligned} c_i' = \quad & (a_1 \vee a_2 \vee b_1) \wedge \\ & (a_3 \vee \overline{b_1} \vee b_2) \wedge \\ & (a_4 \vee \overline{b_2} \vee b_3) \wedge \\ & \cdots \wedge \\ & (a_{k-2} \vee \overline{b_{k-4}} \vee b_{k-3}) \wedge \\ & (a_{k-1} \vee a_k \vee \overline{b_{k-3}}). \end{aligned}$$

3SAT is $\mathcal{NP}$-Complete

$$
\begin{array}{ll}
c_i & (a_1 \vee a_2 \vee a_3 \vee \cdots \vee a_k) \\
\hline
c_i' = & (a_1 \vee a_2 \vee b_1) \wedge \\
 & (a_3 \vee \overline{b_1} \vee b_2) \wedge \\
 & (a_4 \vee \overline{b_2} \vee b_3) \wedge \\
 & \cdots \wedge \\
 & (a_{k-2} \vee \overline{b_{k-4}} \vee b_{k-3}) \wedge \\
 & (a_{k-1} \vee a_k \vee \overline{b_{k-3}}).
\end{array}
$$

- Let us prove that c_i is satisfied if and only if c_i' is satisfiable for some assignment of $b_1, b_2, \ldots, b_{k-3}$.
- If c_i is satisfied for some assignment of $a_1, a_2, \ldots a_k$, then at least one a_j is assigned 1.
- Let us take assignment of $b_1, b_2, \ldots, b_{k-3}$ where $b_1 = 1, b_2 = 1, \ldots, b_{j-2} = 1$ and $b_{j-1} = 0, b_j = 0, \ldots, b_{k-3} = 0$.
- In c_i', all clauses evaluate to 1 because
  - $(a_1 \vee a_2 \vee b_1), (a_3 \vee \overline{b_1} \vee b_2), \ldots, (a_{j-1} \vee \overline{b_{j-3}} \vee b_{j-2})$ evaluate to 1 because $b_1, \ldots, b_{j-2}$ are all 1.
  - $(a_j \vee \overline{b_{j-2}} \vee b_{j-1})$ evaluate to 1 because a_j is 1.
  - $(a_{j+1} \vee \overline{b_{j-1}} \vee b_j), \ldots, (a_{k-2} \vee \overline{b_{k-4}} \vee b_{k-3}), (a_{k-1} \vee a_k \vee \overline{b_{k-3}})$ evaluate to 1 because $\overline{b_{j-1}}, \ldots, \overline{b_{k-3}}, \overline{b_{k-3}}$ are all 1.
- Therefore, c_i' is satisfiable.

3SAT is $\mathcal{NP}$-Complete

$$
\begin{array}{ll}
c_i & (a_1 \vee a_2 \vee a_3 \vee \cdots \vee a_k) \\
\hline
c_i' = & (a_1 \vee a_2 \vee b_1) \wedge \\
 & (a_3 \vee \overline{b_1} \vee b_2) \wedge \\
 & (a_4 \vee \overline{b_2} \vee b_3) \wedge \\
 & \cdots \wedge \\
 & (a_{k-2} \vee \overline{b_{k-4}} \vee b_{k-3}) \wedge \\
 & (a_{k-1} \vee a_k \vee \overline{b_{k-3}}).
\end{array}
$$

- If c_i is not satisfied, then all $a_1, a_2, \ldots, a_k$ are assigned 0.
- In order for c_i' to be satisfied, b_1 has to be 1 (for $(a_1 \vee a_2 \vee b_1)$ to be satisfied)
- b_2 has to be 1 (for $(a_3 \vee \overline{b_1} \vee b_2)$ to be satisfied)...
- b_3 has to be 1 (for $(a_4 \vee \overline{b_2} \vee b_3)$ to be satisfied)...
- ...
- b_{k-3} has to be 1 (for $(a_4 \vee \overline{b_2} \vee b_3)$ to be satisfied)
- But then $(a_{k-1} \vee a_k \vee \overline{b_{k-3}})$ is not satisfied!
- Therefore, there is no way for c_i' to be satisfied if c_i is not satisfied, so c_i' is not satisfiable.
- This shows that c_i is satisfied if and only if c_i' is satisfiable.

3SAT is $\mathcal{NP}$-Complete

Theorem

3SAT is $\mathcal{NP}$-Complete.

Proof

- We can transform any Boolean formula ϕ_1 into formula of the form $c_1 \wedge c_2 \wedge c_3 \wedge \cdots \wedge c_n$, where c_i are all clauses.
- We can also transform every close $c_i = (a_1 \vee a_2 \vee a_3 \vee \cdots \vee a_k)$ (where $k > 3$) into $c'_i = c'_{i_1} \wedge c'_{i_2} \wedge \cdots \wedge c'_{i_{k-1}}$, such that each c'_{i_j} is a clause that has exactly 3 literals.
- Let ϕ'_1 be a formula obtained after transforming ϕ_1 into $c_1 \wedge c_2 \wedge c_3 \wedge \cdots \wedge c_n$ and each c_i into c'_i (if c_i has more than 3 literals).
- It is easy to see that ϕ'_1 is satisfiable if and only if ϕ is satisfiable.

3SAT is $\mathcal{NP}$-Complete

Theorem

3SAT is $\mathcal{NP}$-Complete.

Proof

- The only thing left to do is to transform all the clauses c_i that have less than three literals into conjugations of clauses that have exactly three literals.
- For $c_i = (a_1 \vee a_2)$, we take $c_i' = (a_1 \vee a_2 \vee b_1) \wedge (a_1 \vee a_2 \vee \overline{b_1})$.
- For $c_i = a_1$, we take $c_i' = (a_1 \vee a_1 \vee a_1)$, or we can take

$$(a_1 \vee b_1 \vee b_2) \wedge (a_1 \vee \overline{b_1} \vee b_2) \wedge (a_1 \vee b_1 \vee \overline{b_2}) \wedge (a_1 \vee \overline{b_1} \vee \overline{b_2}).$$

- It is easy to see that in these cases too c_i is satisfied if and only if c_i' is satisfiable (for some assignment of b_1 and b_2).
- Let ϕ_2 be ϕ_1' where all the clauses c_i with less than three literals have been replaced with c_i' in this way.

Therefore, we can transform any Boolean formula ϕ_1 into a 3cnf-formula ϕ_2 such that ϕ_1 is satisfiable if and only if ϕ_2 is satisfiable.

3SAT is $\mathcal{NP}$-Complete

Let

$$\phi_1 = ((x_1 \wedge x_2) \vee (\overline{x_2} \vee x_3 \vee x_4)) \wedge (x_3 \vee \overline{x_1}).$$

Transform ϕ_1 into 3cnf-formula ϕ_2 such that ϕ_1 is satisfiable if and only if ϕ_2 is satisfiable

$$\begin{aligned}
\phi_1 &= ((x_1 \wedge x_2) \vee (\overline{x_2} \vee x_3 \vee x_4)) \wedge (x_3 \vee \overline{x_1}) \\
&= (x_1 \vee \overline{x_2} \vee x_3 \vee x_4) \wedge (x_2 \vee \overline{x_2} \vee x_3 \vee x_4) \wedge (x_3 \vee \overline{x_1}) \\
&= (x_1 \vee \overline{x_2} \vee x_3 \vee x_4) \wedge \\
&\quad (x_3 \vee \overline{x_1}) \\
\phi_2 &= (x_1 \vee \overline{x_2} \vee b_1) \wedge (x_3 \vee x_4 \vee \overline{b_1}) \wedge \\
&\quad (x_3 \vee \overline{x_1} \vee b_2) \wedge (x_3 \vee \overline{x_1} \vee \overline{b_2})
\end{aligned}$$

3SAT is $\mathcal{NP}$-Complete

Theorem

3SAT is $\mathcal{NP}$-Complete.

Proof

- We have shown that 3SAT is in $\mathcal{NP}$.
- We have shown how we can reduce each instance of SAT into an instance of 3SAT.
- This is clearly done in polynomial time, because for each clause of c_i, we generate at most $k+3$ new clauses, where k is the number of literals in c_i.
- Therefore, the number of new clauses will be at most $k+3$ times the number n of literals in ϕ_1 (the size of ϕ_1 instance).
- Since $k \leq n$, there will be $O(n^2)$ clauses, where each one can clearly be generated in polynomial time.
- Therefore, SAT $\propto$ 3SAT and, hence, 3SAT is $\mathcal{NP}$-Complete

CLIQUE is $\mathcal{NP}$-Complete

- We have seen that 3SAT $\propto$ CLIQUE.
- CLIQUE is cleary in $\mathcal{NP}$, therefore the following theorem holds

Theorem

CLIQUE is $\mathcal{NP}$-Complete.

VERTEX-COVER Problem

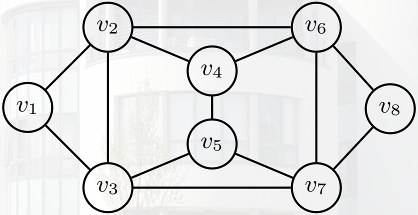
- If G is an undirected graph, a vertex cover of G is a subset of the nodes of G where every edge of G touches one of these nodes.
- The vertex cover is k-node cover if it contains exactly k nodes.

VERTEX-COVER Problem

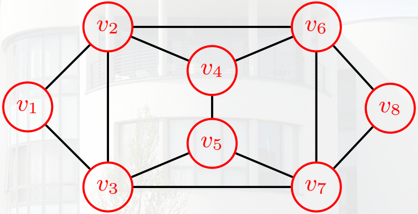
The VERTEX-COVER problem is to determine, for a given graph G and integer number k, whether G contains a k-node cover:

$$\text{VERTEX-COVER} = \{\langle G, k\rangle | G \text{ is an undirected graph that has a } k\text{-node vertex cover}\}.$$

VERTEX-COVER Example

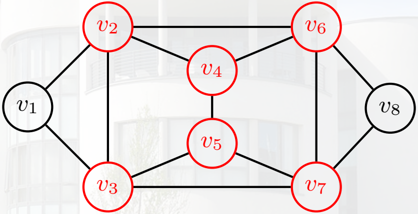


VERTEX-COVER Example



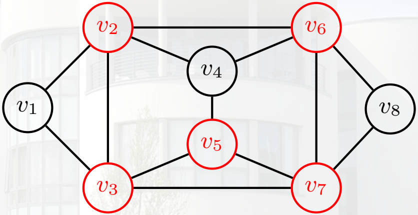
Vertex cover of size 8

VERTEX-COVER Example



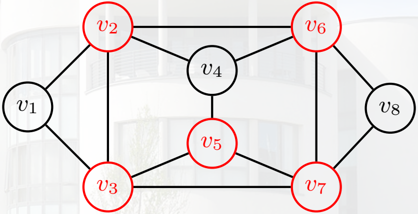
Vertex cover of size 6

VERTEX-COVER Example



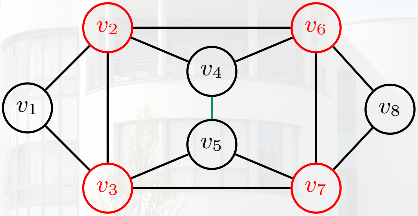
Vertex cover of size 5

VERTEX-COVER Example



There is no vertex cover of size 4!

VERTEX-COVER Example



There is no vertex cover of size 4!

VERTEX-COVER is $\mathcal{NP}$

VERTEX-COVER Problem

The VERTEX-COVER problem is to determine, for a given graph G and integer number k, whether G contains a k-node cover:

$$\text{VERTEX-COVER} = \{\langle G, k\rangle | G \text{ is an undirected graph that has a } k\text{-node vertex cover}\}.$$

- VERTEX-COVER is clearly in $\mathcal{NP}$.
- We can have a NDTM that will, in each path of execution, take a subset of k nodes and easily test in polynomial time whether this is a k-node cover.
- Alternatively, if we guess a solution (a set of k nodes), we can verify in polynomial-time whether that guess is a solution.

VERTEX-COVER is $\mathcal{NP}$-Complete

Theorem (VERTEX-COVER Is $\mathcal{NP}$-Complete)

VERTEX-COVER is $\mathcal{NP}$-Complete.

Proof

- We have seen that VERTEX-COVER is in $\mathcal{NP}$.
- We will show that 3SAT $\propto$ VERTEX-COVER.
- Transformation of the 3cnf-formula ϕ into a graph G and number k such that ϕ is satisfiable if and only if G has a k-node vertex cover is not trivial.
- It, however, demonstrates a commonly-used method of reducing 3SAT to graph problems.

VERTEX-COVER is $\mathcal{NP}$-Complete

Theorem (VERTEX-COVER Is $\mathcal{NP}$-Complete)

VERTEX-COVER is $\mathcal{NP}$-Complete.

- Let ϕ be a 3cnf-formula.
- The graph G that we will transform ϕ to will have two kinds of groups of nodes with edges (or gadgets).

Variable gadgets

- For each variable x in ϕ, we will have a variable gadget of two nodes corresponding to x and $\overline{x}$, with an edge between them.

Clause gadgets

- For each clause in ϕ, we will have a clause gadget of three nodes corresponding to the literals from the clause.
- All the nodes in the clause gadget will be connected by edges.
- Additionally, each literal will be connected by an edge with a corresponding literal from the variable gadget.

3SAT to VERTEX-COVER Example

Let

$$\phi = (x_1 \vee x_1 \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_2}) \wedge (\overline{x_1} \vee x_2 \vee x_2),$$

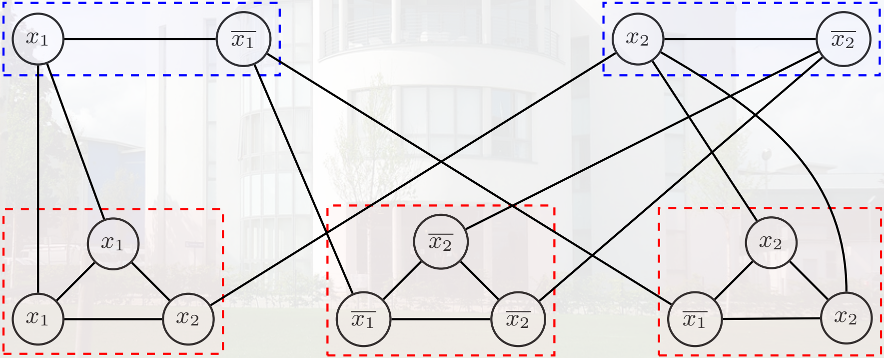
and let us find a graph G that ϕ is transformed into.

3SAT to VERTEX-COVER Example

Let

$$\phi = (x_1 \vee x_1 \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_2}) \wedge (\overline{x_1} \vee x_2 \vee x_2),$$

and let us find a graph G that ϕ is transformed into.



VERTEX-COVER is $\mathcal{NP}$-Complete

Theorem (VERTEX-COVER Is $\mathcal{NP}$-Complete)

VERTEX-COVER is $\mathcal{NP}$-Complete.

- In the graph G constructed in this way, there will be n variable gadgets (where n is the number of variables), each with 2 nodes, totalling $2n$ nodes.
- Note again that a variable is not the same as literal - typically, there will be more literals than variables.
- There will also be m clause gadgets (where m is the number of clauses), each with 3 nodes, totalling $3m$ nodes.
- Therefore, the total number of nodes is $2n + 3m$.

It can be shown that the graph G will have a vertex cover of size $n + 2m$ if and only if ϕ is satisfiable.

Theorem (VERTEX-COVER Is $\mathcal{NP}$-Complete)

VERTEX-COVER is $\mathcal{NP}$-Complete.

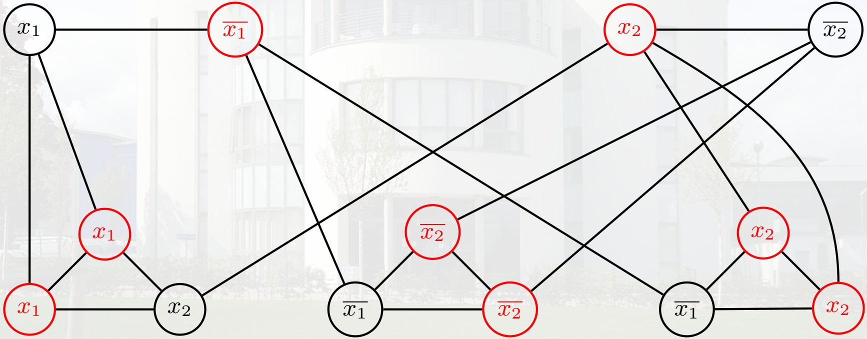
If ϕ is satisfiable, then we choose $n + 2m$ nodes that constitute vertex cover in graph G in the following way:

- In each of the variable gadgets, we choose a node corresponding to x if x has value 1 in the satisfying assignment of ϕ and $\overline{x}$ otherwise.
- In each of the clause gadgets, we choose two nodes that do not correspond to the literal that evaluates to 1 with respect to the satisfying assignment.
- This gives altogether $n + 2m$ nodes (one for each of n variables and two for each of m clauses).

3SAT to VERTEX-COVER Example

$$\phi = (x_1 \vee x_1 \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_2}) \wedge (\overline{x_1} \vee x_2 \vee x_2)$$

is satisfied for $x_1 = 0, x_2 = 1$.



VERTEX-COVER is $\mathcal{NP}$-Complete

Theorem (VERTEX-COVER Is $\mathcal{NP}$-Complete)

VERTEX-COVER is $\mathcal{NP}$-Complete.

Conversely, let there exist a $n + 2m$-node cover in G.

- This cover contains at least one node from each variable gadget.
- It also has to contain at least two nodes from each clause gadget.
- This altogether constitutes $n + 2m$ nodes, so there cannot be two nodes from the same variable gadget in this cover!
- For each variable x, If we assign 1 to x if the vertex cover comprises node corresponding to x and 0 if the vertex cover comprises node corresponding to $\overline{x}$, we will get an assignment that satisfies ϕ.

Therefore, ϕ is satisfiable.

VERTEX-COVER is $\mathcal{NP}$-Complete

Theorem (VERTEX-COVER Is $\mathcal{NP}$-Complete)

VERTEX-COVER is $\mathcal{NP}$-Complete.

- We have seen that VERTEX-COVER is $\mathcal{NP}$.
- We have seen that we can reduce every instance of 3SAT to an instance of VERTEX-COVER (in polynomial-time).
- Therefore, VERTEX-COVER is $\mathcal{NP}$-Complete.

Other $\mathcal{NP}$-Complete Problems

Many other problems have been proven to be $\mathcal{NP}$-Complete

HAMPATH

Does there exist a Hamiltonian path in a given directed graph G between nodes v_i and v_j?

SUBSET-SUM

Given a collection of integer numbers $x_1, x_2, \ldots, x_n$ and a target number t, does there exist a subcollection that adds up to t?

3-COLOURING

Given a graph G, can we colour the nodes of G using three colours (e.g. red, green and blue) such that no two nodes connected by an edge are coloured in the same colour?