Independent Set and Cover Problems

1 Independent Set

INDEPENDENT SET (IS): Given an undirected graph $\mathcal{G}$ and an integer k, does $\mathcal{G}$ contain an independent set of size $\geq k$.

Theorem 1. *IS is **NP**-complete.*

Proof. A two-part proof: (1) IS $\in$ **NP**, (2) IS is **NP**-hard.

(1). IS $\in$ **NP**

Certificate: A proposed set S.

Certifier: A polytime algorithm which tests that S is independent and $|S| \leq k$.

(2). IS is **NP**-hard

We reduce from a known **NP**-hard problem (3SAT) to IS.

Theorem 2. *3SAT $\leq p$ IS.*

Proof. The input to the reduction is a 3SAT formula F, and the output should be a graph $\mathcal{G}$ and integer k. It must run in polytime, and satisfy:

(1). If $\mathcal{G}$ has an IS of size $\geq k$, then F is satisfiable.

(2). If F is satisfiable, then $\mathcal{G}$ has an IS of size $\geq k$.

A useful definition of 3SAT here is that for each clause C_j, we pick a true literal $l_{j,i}$ such that our selection never includes both x_i and $\bar{x_i}$. Our choice in 3SAT is which designated literals to pick, which will map to our choice in IS of which vertices to include. For each C_j and $l_{j,i}$, we have one node $v_{j,i}$ in $\mathcal{G}$. Then, we set $k = m$ (number of vertices to number of clauses). We add edges between the $v_{j,i}$ in a fixed j across all i (which forces picking only one node in a clause). This reduction is obviously polytime, so now we prove correctness.

(1). Assume $\mathcal{G}$ has an IS S of size $\geq m$. For each $v_{j,i} \in S$, make $l_{j,i}$ true, and vice versa. Because of the variable edges, we never try to make x_i and $\bar{x_i}$ true. This strategy must satisfy the clauses because $|S| \geq m$, and it has at most one node per C_j, so it must have at least one literal per clause. Thus, F is satisfied.

(2). If F is satisfiable, then let $\hat{i}(j)$ be a true literal in C_j under a fixed satisfying assignment. Let $S = \{v_j, \hat{i}(j)\}$. Then $|S| = m$ and S is independent because it does not violate the clause edges (we included only one node per clause) or variable edges (only one of x_i and $\bar{x_i}$ can be selected in the assignment). $\square$

Since we proved that IS is both in **NP** and **NP**-hard, IS must be **NP**-complete. $\square$

2 Vertex Cover

Let a vertex cover be a set $S \subseteq V$ such that for all edges e, at least one endpoint of e is in S. VERTEX COVER (VC): Given a graph $\mathcal{G}$ and integer k, is there a vertex cover of size $\leq k$?

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

Proof. A two-part proof: (1) VC $\in$ **NP**, (2) VC is **NP**-hard.

(1). VC $\in$ **NP**

Certificate: A set S that supposedly covers all edges.

Certifier: A polytime algorithm which compares $|S| \leq k$ and ensures S covers all edges.

(2). VC is **NP**-hard

We reduce from a known **NP**-hard problem (IS) to VC.

Theorem 4. *IS $\leq p$ VC*

Proof. The input to the reduction is a graph $\mathcal{G}$ and integer k; the output is a graph $\mathcal{G}'$ and integer k'. We notice that IS and VC are complements – that is, S is an independent set if and only if $\bar{S}$ is a vertex cover. Thus, $\mathcal{G}$ has an independent set of size $\geq k$ if and only if $\mathcal{G}$ has a vertex cover of size $\leq n-k$. Then, we can just set $\mathcal{G}' = \mathcal{G}$ and $k' = n-k$, which obviously runs in polytime. □

Since we proved that VC is both in **NP** and **NP**-hard, VC must be **NP**-complete. □

3 Set Cover

SET COVER (SC): Given elements E, subsets $S_1, S_2, \ldots, S_m \subseteq E$, and an integer k, is there a set $T \subseteq \{1 \ldots m\}$ with $|T| \leq k$ and $\bigcup_{i \in T} S_i = E$? This problem is equivalent to figuring out the minimum number of routers one would need to cover every office in a building, etc.

Theorem 5. *SC is **NP**-complete.*

Proof. A two-part proof: (1) SC $\in$ **NP**, (2) SC is **NP**-hard.

(1). VC $\in$ **NP**

Certificate: T.

Certifier: A polytime algorithm which compares $|T| \leq k$ and checks if $\bigcup_{i \in T} S_i = E$.

(2). VC is **NP**-hard

We reduce from a known **NP**-hard problem (VC) to SC.

Theorem 6. *VC $\leq p$ SC*

Proof. The input to the reduction is a graph $\mathcal{G}$ and integer k; the outputs are elements E, sets $S_1, \ldots S_m \subseteq E$, and an integer k'. We have one set S_v for each $v \in V$ containing all edges incident on (and thus covered by) $v \in \mathcal{G}$. Usually we would prove this more formally, but we were running out of time, so we determined that E is equivalent to the set of edges in $\mathcal{G}$, while $k' = k$. This algorithm is obviously polytime, and it is correct because $\bigcup_{v \in T}$ edges covered by $v = \bigcup_{v \in T} S_v$, so vertex covers in $\mathcal{G}$ and set covers are essentially the same. □

Since we proved that SC is both in **NP** and **NP**-hard, SC must be **NP**-complete.

An important realization is that we reduce from a specific problem to a more general one; 3SAT is actually an extremely specific problem, while SC is more general. While we usually move in that direction, all **NP**-complete problems are equally hard, so we can reduce from SC to 3SAT via using CERT to simulate them. □

4 Some famous NP-complete problems

Logic: SAT, 3SAT, X3SAT, NAE3SAT, ...

Graphs: IS, VC, DOMINATING SET, GRAPH COLORING, ...

Sets: SC, SET PACKING, ...

Other: PARTITION, SUBSET SUM, KNAPSACK, STEINER TREE, HAMILTONIAN CYCLE, TRAVELING SALESMANs