1 Reduction from 3-SAT to VC

We started with an example based on the 3-SAT instance $(a \lor b \lor \neg c) \land (\neg a \lor \neg b \lor c) \land (\neg a \lor b \lor d)$. See visual aid.

In general, the reduction maps an instance $\phi$ of 3-SAT to an instance $(G,k)$ of vertex cover in the following way. Every variable in $\phi$ is mapped to a “variable gadget” which consists of two vertices joined by an edge; one vertex is labeled with the variable and the other with its negation. Every clause in $\phi$ is mapped to a “clause gadget” which consists of a clique of three vertices, each labeled with a literal from the respective clause. Then $G$ is constructed to consist of all variable and clause gadgets, with an edge from each literal in each variable gadget to all instances of that literal in the clause gadgets. The variable $k$ is set to be $n + 2m$ where $n$ is the number of variables in $\phi$ and $m$ is the number of clauses in $\phi$.

Now let’s prove that the reduction is correct, that $3\text{-SAT} \leq^P_m \text{VC}$. First we show that a “yes” instance of 3-SAT maps to a “yes” instance of vertex cover. If the 3-SAT formula is satisfiable, that means that there is a truth assignment to each variable in the formula that makes each clause true. We choose the vertex corresponding with the true literal in each variable gadget to be in the vertex cover. Since each clause has at least one true literal, it follows that every clause gadget has at least one incoming edge already covered, and therefore the other edges adjacent to that gadget can be covered by adding in two more vertices per clause (this also covers the triangle within each clause gadget). Thus we know that if $\phi$ is a “yes” instance of 3-SAT then $(G,k)$ is a “yes” instance of vertex cover.

Now, conversely, assume that we have a vertex cover of size $n + 2m$ where $n$ is the number variables and $m$ is number of clauses. If we had a vertex cover with $n + 2m$ vertices in it, then $n$ of them are expended in the variable gadgets (we need to cover the edge in each variable gadget). Similarly, for each clause gadget, we must choose at least 2 of the vertices to be in the vertex cover in order to cover every edge of the triangle. We claim that a satisfying assignment corresponding to the vertex cover (i.e. choose the literals in $\phi$ to be true or false, matching the vertex cover) comprises a satisfying assignment of $\phi$. Each clause must have a vertex which is connected to a variable gadget literal which is true (by virtue of that edge being covered). This corresponds directly to $\phi$ having at least one true literal in each clause.

We shown that the constuction induces a mapping reduction $3\text{-SAT} \leq^P_m \text{VC}$, and it’s straightforward to see that the transformation from $\phi$ to $(G,k)$ can be computed in polynomial time, so we actually have $3\text{-SAT} \leq^P_m \text{VC}$. To conclude that VC is NP-complete, we must verify that VC is NP-complete. Each branch of a nondeterministic machine can consider a different truth assignment to the variables. That would be $2^n$ different assignments, and we can create $2^n$ branches in time $n$ on such a machine. If any one of these branches returns true, the whole machine returns true, which is correct (it’s a “yes” instance if and only if there’s at least one truth assignment that makes the whole thing true). If it’s a “no” instance, all branches will return false, which makes the whole machine return false, which is correct. We conclude that VC is NP-complete.