CSE491/596, Mon. 10/23/23: Second Lecture on NP-Completeness

Picking up the **Independent Set** example, here is how we frame and state the reduction. There are three parts which I call "**Construction**", "**Complexity**" (often short), and "**Correctness**" of the reduction.

Given any 3CNF formula $\phi(x_1, \dots, x_n) = C_1 \wedge C_2 \wedge \cdots \wedge C_m$, we build a graph $G = (V, E)$ and set $k = m + n$ to get an equivalent instance (G, k) of **Independent Set** as follows:

- V consists of $2n$ "rung nodes" labeled $x_1, \bar{x}_1, x_2, \bar{x}_2, \dots, x_n, \bar{x}_n$ and (up to) $3m$ "clause triangle nodes". (It is exactly $3m$ nodes if ϕ is in "strict 3CNF", which you are allowed to assume.)
- E first has n "rung edges", each between some x_i and its negation $\bar{x}_i$.
- Then E has $3m$ "clause gadget edges" to make a triangle for each clause.
- Finally and most critically, E has $3m$ "crossing edges". For each occurrence of a positive literal x_i in a clause gadget, the edge goes to the negated $\bar{x}_i$ in its "rung". (The edges and whole graph are undirected.) For each occurrence of a negative literal $\bar{x}_i$ in a clause gadget, the edge goes to the positive x_i in its "rung".

This finishes the contruction of $G = (V, E)$ and k in general.

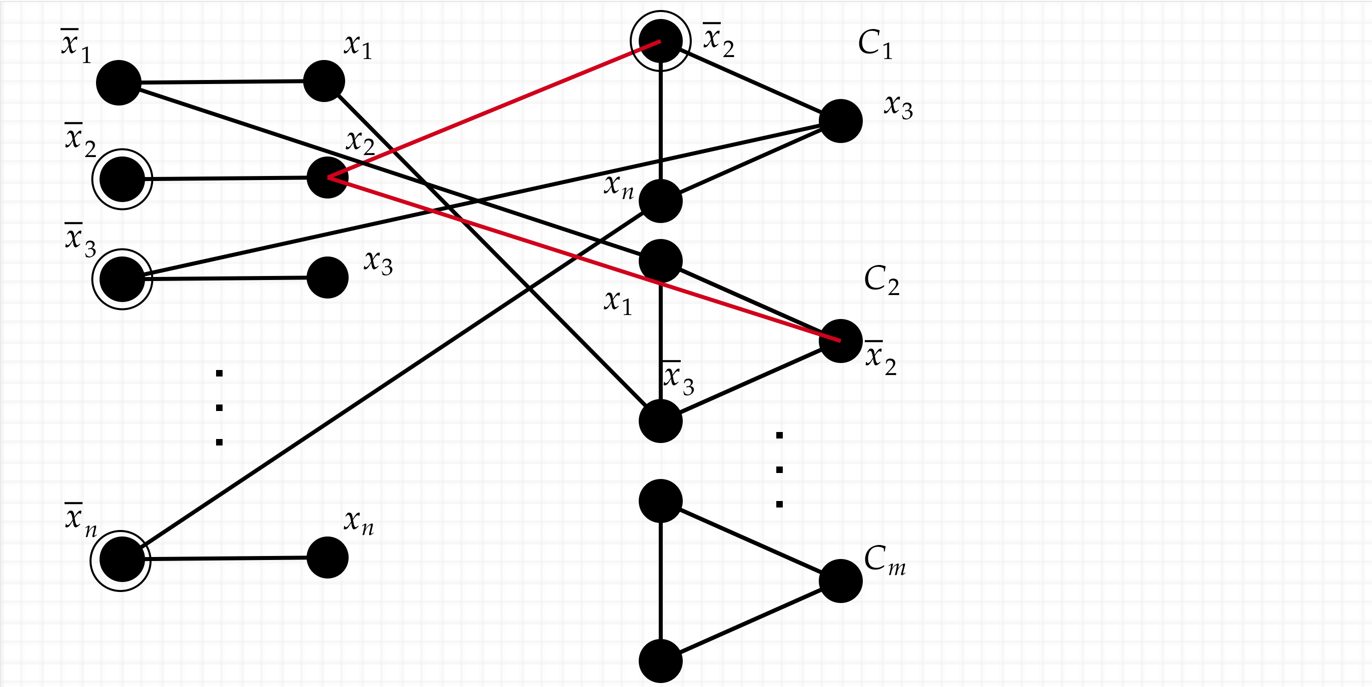
Complexity: G can be build with simple passes over ϕ. [Usually this can be done in a sentence or two.]

Correctness: [This takes time and care...To be fully safe, show both of these implications:

- **If** ϕ has a satisfying assignment $a = (a_1, a_2, \dots, a_n)$, **then** from a we can make choices of the existentially questioned object (in this case, an independent set) to meet the stated requirements (here, including meeting the size target k).
- **If** there is an object (i.e., "witness") that answers "yes" to the problem, **then** from that object we can find a satisfying assignment to ϕ.

Together, these show that ϕ is satisfiable $\iff$ the answer to the target instance (G, k) is "yes". This completes the requirements of reducing 3SAT to the target problem (by a polynomial-time many-one reduction), so the target problem is NP-hard. Since it belongs to NP, it is NP-complete.]

For this reduction, we make the "rungs" into actual edges between each x_i and its negation $\bar{x}_i$ and give each clause three nodes to make a triangle. Each clause node is labeled by a literal in the clause. Later we will include the clause index j, not just the variable index i, when identifying this *occurrence* of the literal in a clause to define V as a set, where $G = (V, E)$.

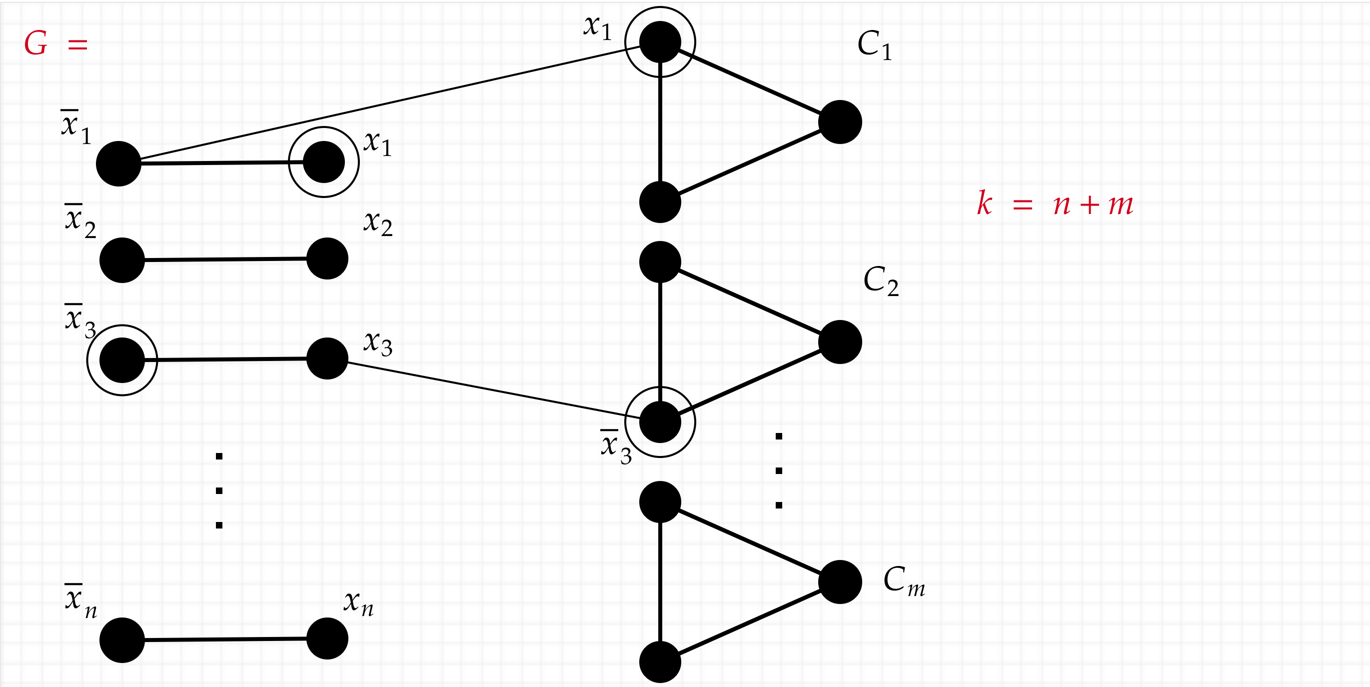


$\phi = (x_3 \vee \overline{x}_2 \vee x_4) \wedge (x_1 \vee \overline{x}_2 \vee \overline{x}_3).$

The immediate effect, even before we consider an example of a formula, is that the maximum possible k for an independent set S in the graph G is $n + m$. The most one can do is take one vertex from each rung and one from each triangle to make S. Note that the vertices chosen from each rung specify a truth assignment to the variables.

The final goal of the reduction is to add a third set of edges, which I call "crossing edges", to enforce that a set S of size $n + m$ is possible if and only if its corresponding assignment satisfies the formula. The basic idea, even before we consider a formula, is as follows.

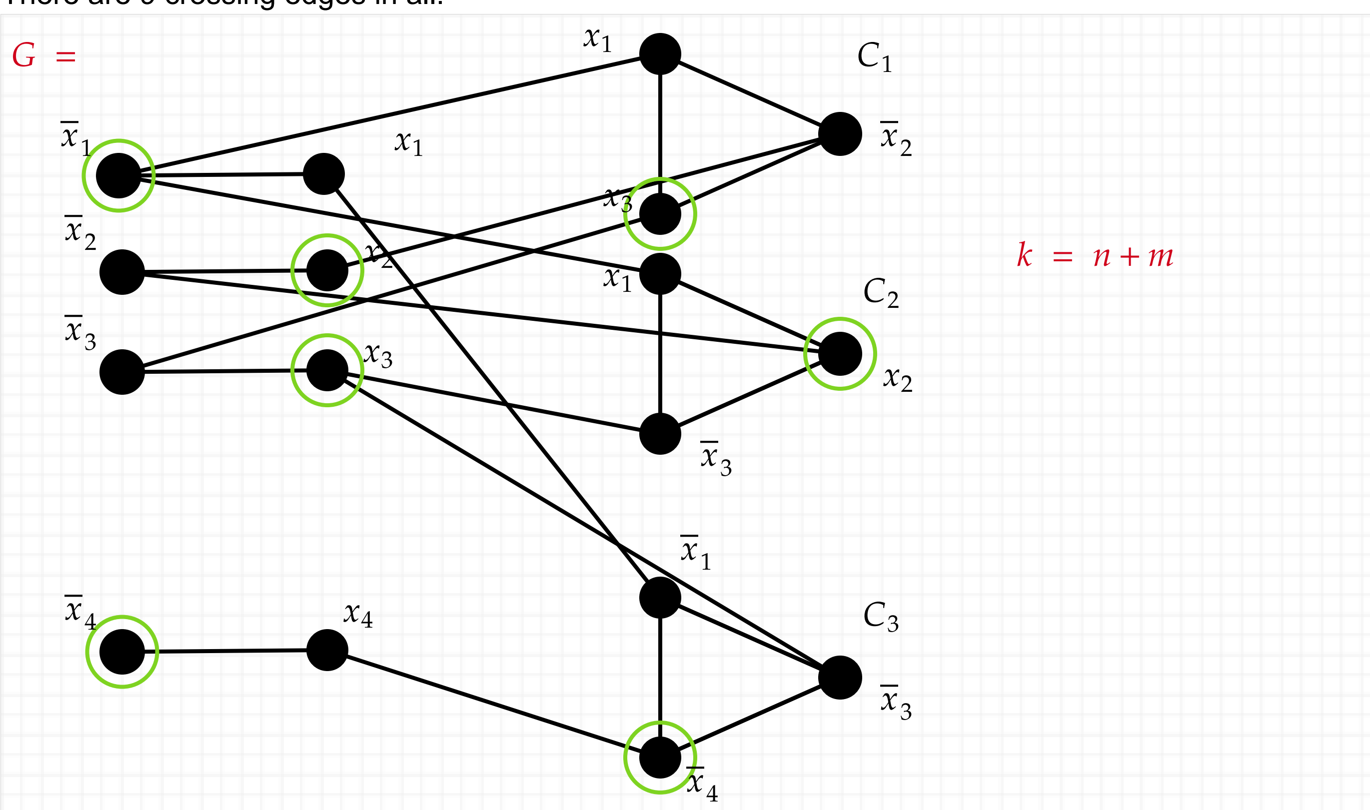
- Suppose clause C_1 includes the positive literal x_1. Then we connect a crossing edge from x_1 in C_1 to the *opposite* literal $\overline{x}_1$ in the rung.
- Suppose clause C_2 includes the negated literal $\overline{x}_3$. Then we connect a crossing edge from $\overline{x}_3$ in C_2 to the opposite literal in the rung, which is just x_3.



The edges ensure that choosing a satisfied literal in each clause will not conflict with the truth assignment. Here is an example formula.

$$\phi = (x_1 \vee \overline{x}_2 \vee x_3) \wedge (x_1 \vee x_2 \vee \overline{x}_3) \wedge (\overline{x}_1 \vee \overline{x}_3 \vee \overline{x}_4).$$

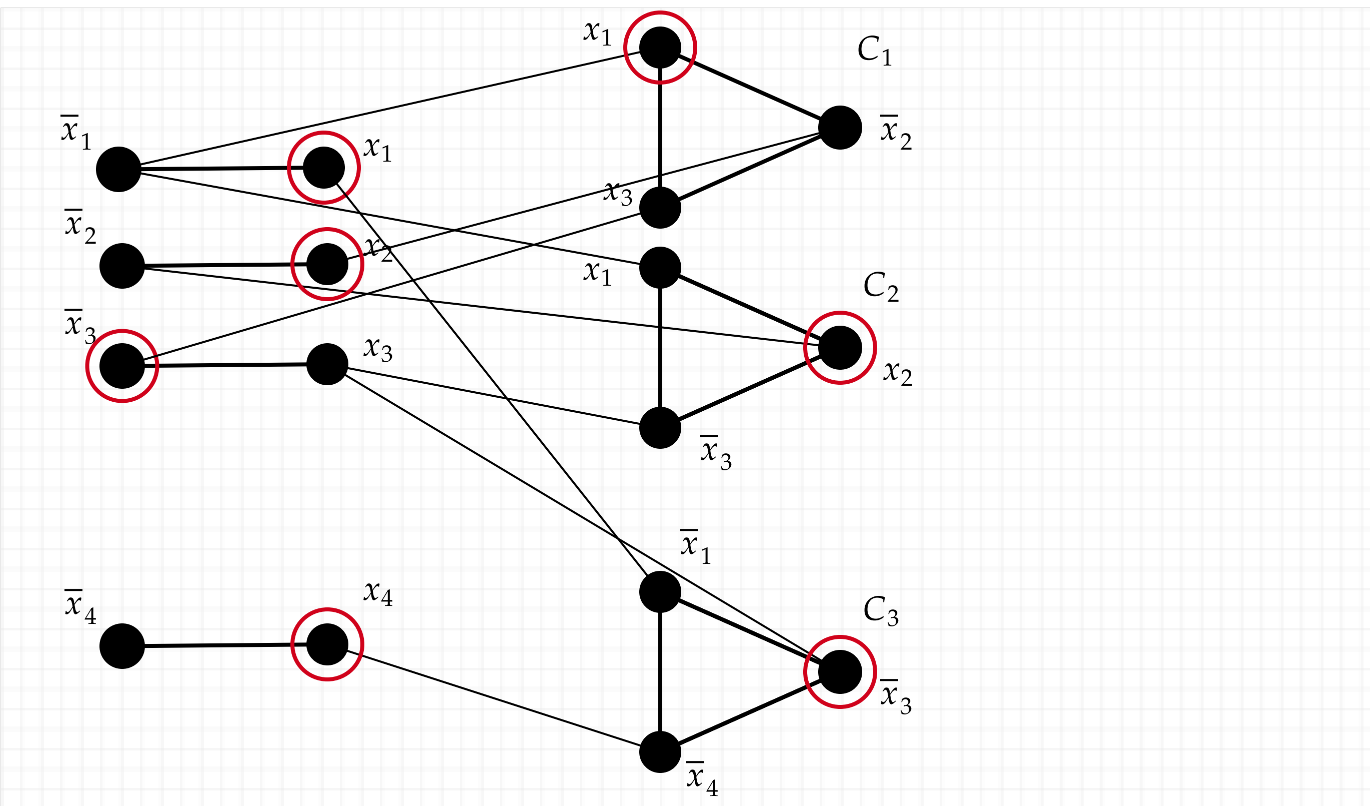
There are 9 crossing edges in all:



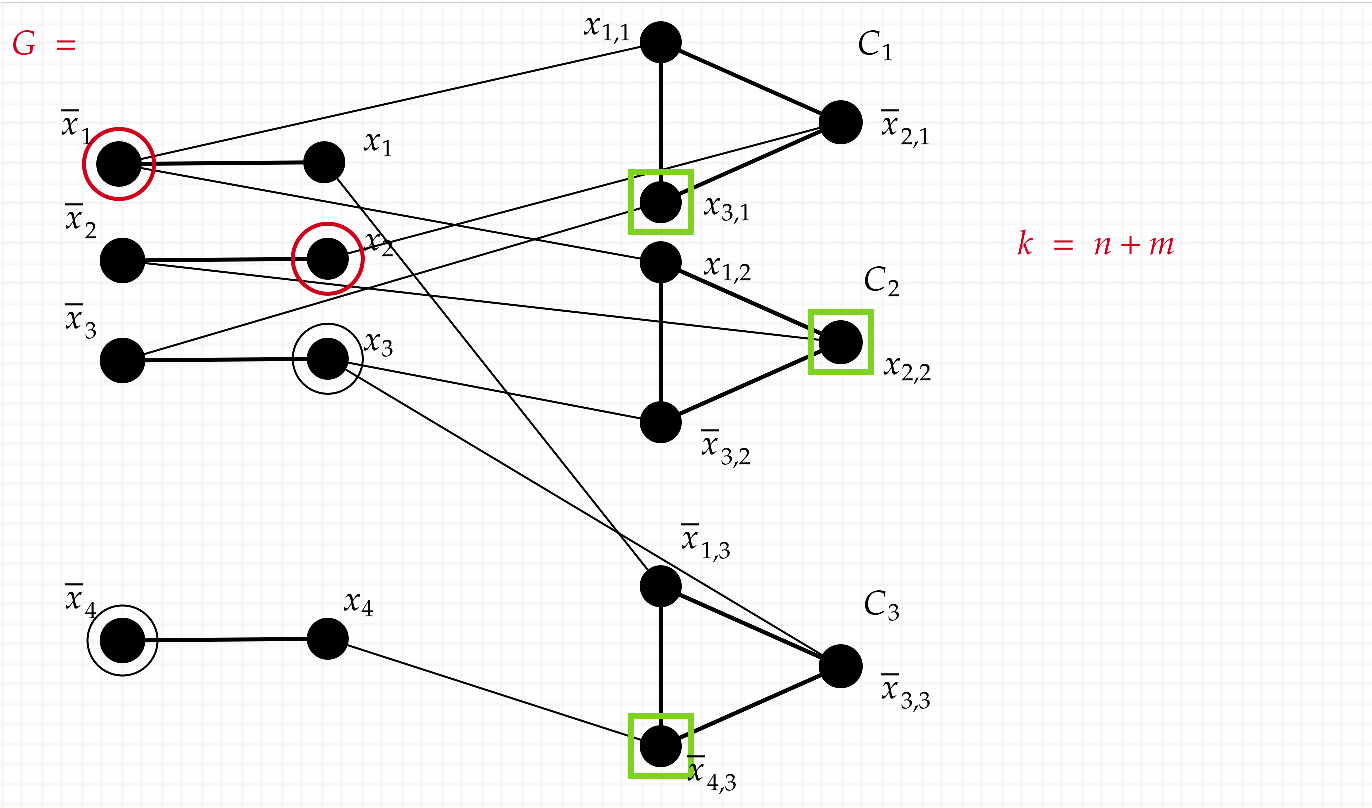
Note that a choice of vertices for S is not part of G---not part of the reduction function f itself. It is only part of the analysis of why the reduction is correct.

To illustrate the analysis, note that the example formula ϕ is satisfiable. In fact, it has many satisfying assignments. (To make a strict 3CNF formula that is unsatisfiable and not use trivialities like duplicate literals in the same clause, one needs to have at least 8 clauses.) For $a = 1101$ and
$\phi = (x_1 \vee \overline{x}_2 \vee x_3) \wedge (x_1 \vee x_2 \vee \overline{x}_3) \wedge (\overline{x}_1 \vee \overline{x}_3 \vee \overline{x}_4)$,
one of them is to set x_1 true and x_3 false; then x_2 and x_4 become "don't-cares":

[2023 Note: Rather than jump between diagrams that were based on how I lectured in previous years---including once with 80-minute Tue.+Thu. lectures---what I did was stay longer with one diagram and spend more time moving and copying the choice-making rings around the nodes. So what follows does not exactly represent how I lectured, but it shows much the same things.]



Or we can try setting x_1 false and x_2 true:
$\phi = (x_1 \vee \overline{x}_2 \vee x_3) \wedge (x_1 \vee x_2 \vee \overline{x}_3) \wedge (\overline{x}_1 \vee \overline{x}_3 \vee \overline{x}_4)$,



This blocks two of the literals in C_1. We have to set x_3 true. This blocks $\overline{x}_3$ in C_2 and x_1 is already blocked there. Luckily we can choose x_2 in C_2. Since we already have $\overline{x}_1$ as an option in C_3 (but not $\overline{x}_3$), and variable x_4 is not connected elsewhere, it is again a don't care.

One other thing happened in the diagram: each clause node added a subscript for the clause. This enables us to define the reduction formally by specifying the graph in set notation. [Well, in lecture this time, in 2023, I said this much subscripting was yucky and would be unnecessary.]

$V = \{x_i, \overline{x}_i : 1 \le i \le n\} \cup \{x_{ij} : C_j \text{ has } x_i\} \cup \{\overline{x}_{i,j} : C_j \text{ has } \overline{x}_i\}$
$E = E_{rungs} \cup E_{clauses} \cup E_{crossing}$
$E_{rungs} = \{(x_i, \overline{x}_i) : 1 \le i \le n\}$
$E_{clauses} = \{(x_{i,j}, x_{k,j}) : C_j \text{ has } x_i \text{ and } x_k\} \cup \{(\overline{x}_{i,j}, \overline{x}_{k,j}) : C_j \text{ has } \overline{x}_i \text{ and } \overline{x}_k\}$
$\cup \{(x_{i,j}, \overline{x}_{k,j}) : C_j \text{ has } x_i \text{ and } \overline{x}_k\}.$

[Side Q: Do we need to add "$\cup \{(\overline{x}_{i,j}, x_{k,j}) : C_j \text{ has } \overline{x}_i \text{ and } x_k\}$? No: things are symmetric.]

$E_{crossing} = \{(x_i, \overline{x}_{i,j}) : \overline{x}_i \in C_j\} \cup \{(\overline{x}_i, x_{i,j}) : x_i \in C_j\}.$

And, of course, $k = n + m$ completes the definition of the reduction function $f(\phi) = (G, k)$. The one benefit of laying out these sets is that they show exactly how to *compute* the graph, and how big it gets. We have $|V| = 2n + 3m$ and $|E| = n + 3m + 3m = n + 6m$. Both are in fact *linear* in the size order-$(n+m)$ of ϕ. The edge lists can be streamed in one pass through the variables and clauses. [Note that although I have not settled on any one formal definition of "streaming algorithm", the idea of them is useful to sharpen the understanding of how the reductions are efficiently computable.] This is indeed a quasilinear-time (**DQL**) reduction.

So we have given the **C**onstruction, shown that its **C**omplexity is well within polynomial time, so it remains to show **C**orrectness: $\phi \in 3SAT \iff f(\phi) \in INDSET$. That is, we need to show

the 3CNF formula ϕ is satisfiable $\iff G$ has an independent set S of size $k = m+n$ (the max possible size)

($\implies$): Suppose a satisfies ϕ. Form S by taking the n rung nodes set true by a and choosing one node from each clause that is satisfied. Then by similar reasoning about the crossing edges, S is an independent set of size $n+m$ in G. [Note that even after fixing a, where you've made choices also for "don't-care" variables, there may be multiple S sets because two or three nodes might be satisfied in any given clause. So it is not a 1-to-1 correspondence. But it does have the property Levin cared about, which is that a choice of S uniquely identifies a satisfying assignment.]

($\impliedby$): Given S, it has exactly n nodes from rungs and one node from each clause. For each i, S has either x_i or $\overline{x}_i$. The choices determine a unique truth assignment a. Now consider any clause C_j and let $x_{i,j}$ or $\overline{x}_{i,j}$ be the label of the node chosen. In the former case, there is a crossing edge from $x_{i,j}$ to $\overline{x}_i$. Now $\overline{x}_i$ cannot be the node in S from the i-th rung because that would give S a clash. So the rung node in S must be x_i, so the corresponding assignment makes x_i true, and that satisfies the clause C_j. In the latter case, there is a crossing edge from $\overline{x}_{i,j}$ to x_i. Now x_i cannot be the node in S from the i-th rung because that would give S a clash. So the rung node in S must be $\overline{x}_i$, so the corresponding assignment makes x_i false. Since C_j has $\overline{x}_{i,j}$ in this case, that likewise satisfies the clause C_j. Since C_j is arbitrary, this means a satisfies ϕ. ☒

Since $IND\ SET \leq_m^p CLIQUE$ and $IND\ SET \leq_m^p VERTEX\ COVER$ these problems (which we showed to be in NP) are also NP-complete.