CS 3510 Algorithms 10/30/2023

Lecture 16: Independent Set, Clique, and Vertex Cover

Lecturer: Abrahim Ladha *Scribe(s): Richard Zhang*

Let's look at some reductions and discover some more problems that are NP-complete. We will be looking at Independent Set, Clique, and Vertex Cover.

1 Independent Set

For a given graph with vertex set V and edge set E, a subset of vertices $I \subseteq V$ is independent if there is no edge between any two vertices in the subset. Lets look at some examples where finding an independent set is simple.

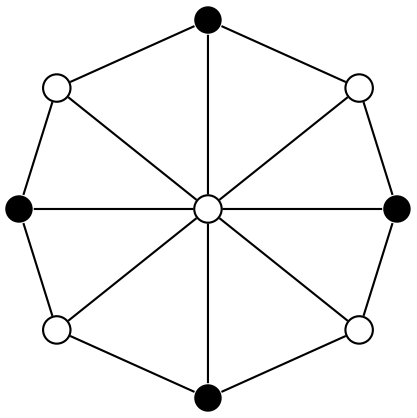

Figure 1: Independent set of Wheel graph

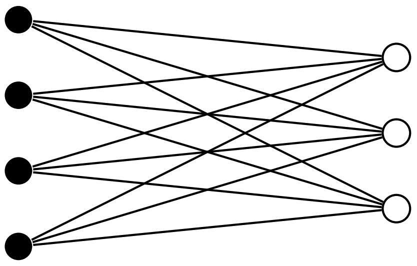

Figure 2: Independent set of Bipartite Graph

In the graphs from Figure 1 and 2, the black vertices would form an independent set since there is no edge that has both ends as black vertices. However, finding an independent set for a graph in general is a really hard problem and is in fact an NP-complete problem, which we will show now. Let formalize it as a decision problem and define it as the following:

$$\texttt{INDEPENDENT-SET} = \{\langle G, g\rangle \mid G \text{ has an independent set of size} g\}$$

First, we will show that INDEPENDENT-SET $\in$ NP. We will define a verifier that takes in as input a problem $\langle G, g\rangle$ and a witness $\langle v_1, ..., v_k\rangle$, where G is a graph, g is an integer, and $\langle v_1, ..., v_k\rangle$ is a subset of vertices. We first check that $k = g$. We then check for each pair of distinct vertices $v_i, v_j \in \{v_1...v_k\}$ that there does not exist an edge between them. If both of these checks pass, then we can return true (the problem instance $\langle G, g\rangle \in$ INDEPENDENT-SET). Else, we return false. Since these checks take polynomial time, we have a polynomial time verifier and can therefore conclude that INDEPENDENT-SET $\in$ NP.

Now, we will show that INDEPENDENT-SET is NP-hard. We will reduce from the $3SAT$ problem. So far, all of our reductions have been from boolean formula problems. How can we convert a CNF-formula to a graph? This does not seem too obvious.

Let's delve into the 3SAT problem. When you have a satisfying assignment of a formula, you cannot select literals x and $\bar{x}$ to be true simultaneously. For example:

$$(a \vee x \vee b) \wedge (c \vee \bar{x} \vee d)$$

Choosing x in the first clause turns off $\bar{x}$ in the second. We want to do something similar in a graph. Choosing a vertex to be part of a candidate independent set turns off its neighbors from being included in the set.

From the idea above, we can build a polynomial time reduction $f : \varphi \to \langle G, g\rangle$ such that

$$\varphi \in \texttt{3SAT} \iff \langle G, g\rangle \in \texttt{INDEPENDENT-SET}$$

f will take in as input φ, a formula in 3CNF. For each clause in φ, create one "triangle" per clause where every vertex in the clause is unique, the vertices are labeled by the clause's literals, and these vertices are connected to each other [1]. If a literal x is a vertex in a triangle, put an edge to $\bar{x}$ in all the other triangles. Lets denote this generated graph as G. We then set g to be the number of clauses in φ. We now return this new problem $\langle G, g\rangle$. Note that this reduction is computable in polynomial time. Building G takes linear time to build the triangles and, at the worst case, $O(n^2)$ time for building the other pairs of edges.

Let's look at an example. If $\varphi = (\bar{x} \vee y \vee \bar{z}) \wedge (x \vee \bar{y} \vee z) \wedge (x \vee y \vee z) \wedge (\bar{x} \vee \bar{y})$, then the graph G would look like the one presented in Figure 3.

[1] For a clause with 2 variables, it would be two vertices connected by an edge. For a clause with 1 variable, it would simply be the vertex itself

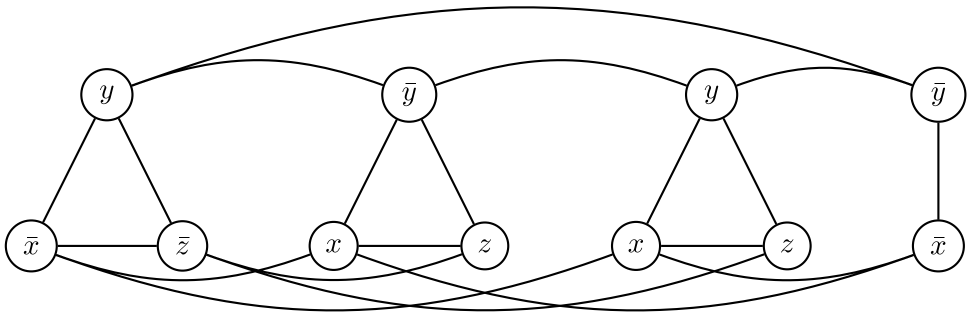

Figure 3: Constructed graph of formula $(\bar{x} \vee y \vee \bar{z}) \wedge (x \vee \bar{y} \vee z) \wedge (x \vee y \vee z) \wedge (\bar{x} \vee \bar{y})$

If $\varphi \in$ `3SAT`, then there exists a satisfying assignment of the variables that makes φ true. For each clause, select the corresponding vertex of one of the literals that would be evaluated to true in the satisfying assignment of the formula. These selected vertices would form an independent set since only one literal is selected per "triangle" of each clause and that it is impossible for a literal and its negation to be true and to be selected. This results in no edges having both ends being selected. Since one vertex of each clause will be selected, the size of the independent set will be equal to g (which was set to the number of clauses in the reduction). Therefore, $\varphi \in$ `3SAT` $\implies \langle G, g \rangle \in$ `INDEPENDENT-SET`.

If $\langle G, g \rangle \in$ `INDEPENDENT-SET`[2], then an independent size of size g exists. By construction, there is g "triangles" in the graph, and since you cannot pick two (or more) from a single triangle, this independent set must have one vertex exactly in each triangle. Each of the selected vertices corresponds to the assignment of each clause in the formula φ, so we see that $\langle G, g \rangle \in$ `INDEPENDENT-SET` $\implies \varphi \in$ `3SAT`.

Since we found a valid polynomial reduction, we can conclude that `3SAT` $\leq_p$ `INDEPENDENT-SET` and therefore `INDEPENDENT-SET` is NP-hard.

Since `INDEPENDENT-SET` $\in$ NP and `INDEPENDENT-SET` is NP-hard, we conclude that `INDEPENDENT-SET` is NP-complete. Now that we have one graph problem which is NP-complete, it will make more graph NP-complete problems easier to prove.

2 Clique

A **complete graph** is a graph where every vertex is connected to every other vertex. A **clique** is a complete subgraph of a graph, meaning that every vertex in this subgraph is connected to every other vertex in the subgraph. Figure 4 has examples of complete graphs. Note that Kn denotes a complete graph with n vertices.

[2]To clarify, this is a G that was constructed from the reduction, not some general graph.

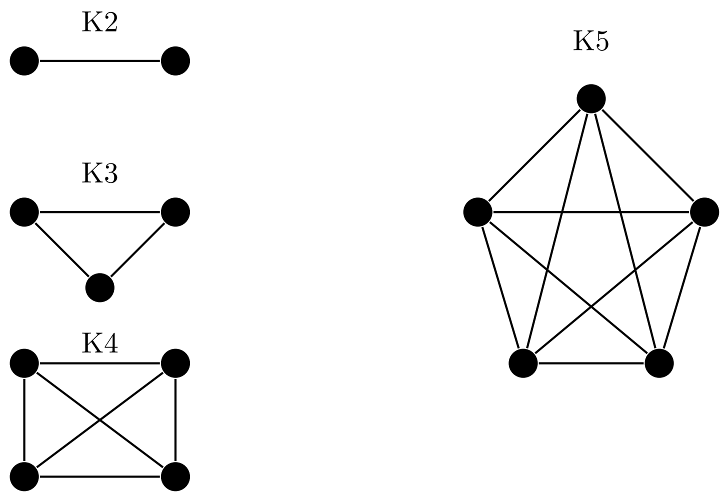

Figure 4: Complete Graphs K2, K3, K4, and K5

Figure 5 presents an example of a clique of size 3 (as indicated by the black nodes) present in a graph.

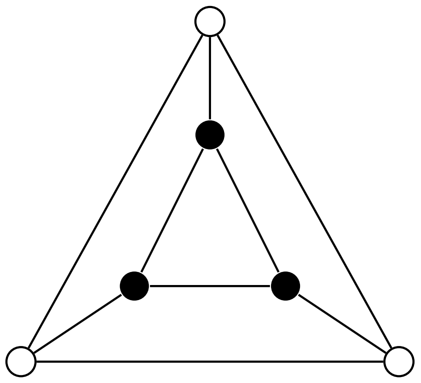

Figure 5: Clique of a graph

Determining if a clique of a certain size exists in a graph is a very hard problem and is in fact NP-complete. To prove this, let formalize it as a decision problem and define it as the following:

$$\texttt{CLIQUE} = \{\langle G, g\rangle \mid G \text{ has a clique of size } g\}$$

First, we will show that $\texttt{CLIQUE} \in \mathsf{NP}$. We will define a verifier that takes in as input a problem $\langle G, g\rangle$ and a witness $\langle v_1, ..., v_k\rangle$, where G is a graph, g is an integer, and $\langle v_1, ..., v_k\rangle$ is a subset of vertices. The verifier checks that $\{v_1...v_k\}$ has size g. Next, it checks if there exists an edge between every distinct pair $v_i, v_j \in \{v_1...v_k\}$. If both of these conditions pass, then our witness $\langle v_1, ..., v_k\rangle$ is a clique. This verifier takes polynomial time, so we can conclude that $\texttt{CLIQUE} \in \mathsf{NP}$.

Now, we will show that `CLIQUE` is NP-hard by reducing not from `3SAT` but from `INDEPENDENT-SET`. We need to find a reduction $f : \langle G, g\rangle \rightarrow \langle \bar{G}, g\rangle$ such that

$$\langle G, g\rangle \in \texttt{INDEPENDENT-SET} \iff \langle \bar{G}, g\rangle \in \texttt{CLIQUE}$$

Lets make $\bar{G}$ the **complement graph** of G and make the variable g stay the same. The complement of a graph $\bar{G}$ includes the edges that do not exist in G and excludes the edges that do exist in G. More precisely, $\bar{G}$ has the same vertex set as G and an edge set that contains the unordered pairs of the vertices that do not exist in the original edge set of G. This reduction is polynomial since we are simply looping through pairs of vertices to find the edges that do and do not exist. Figure 6 presents an example of the complement of the graph.

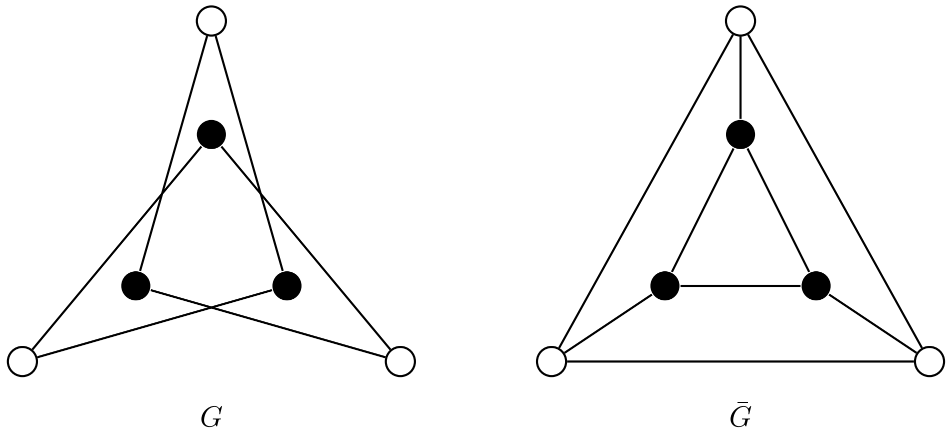

Figure 6: Complement graph

Notice that if a set of vertices are independent, then they share no edges. However, in the complement graph, these same set of vertices have every possible edge between two vertices in the set.

If $\langle G, g\rangle \in$ `INDEPENDENT-SET`, then there exists a selection of the vertices of size g with no edges between then. Then if an edge e is not in the edge set of G, it must be in the edge set of $\bar{G}$. Thus, those same vertices in $\bar{G}$ share all edges. That is the definition of a clique, so $\bar{G}$ has this clique of size g. Therefore, we can conclude that $\langle G, g\rangle \in \texttt{INDEPENDENT-SET} \implies \langle \bar{G}, g\rangle \in \texttt{CLIQUE}$.

Note that since the complement of $\bar{G}$ ($\bar{\bar{G}}$) is equal to G itself, this is a rare time we get the reverse argument for free. If $\langle \bar{G}, g\rangle \in$ `CLIQUE`, then the clique of size g in $\bar{G}$ is an independent set of size g in G. Therefore, we can conclude that $\langle \bar{G}, g\rangle \in \texttt{CLIQUE} \implies \langle G, g\rangle \in \texttt{INDEPENDENT-SET}$.

Since we found a valid polynomial reduction, we can conclude that $\texttt{INDEPENDENT-SET} \leq_p \texttt{CLIQUE}$ and therefore `CLIQUE` is NP-hard.

Since CLIQUE $\in$ NP and CLIQUE is NP-hard, we conclude that CLIQUE is NP-complete.

3 Vertex Cover

A **vertex cover** is a selection of vertices such that every edge shares one end in the cover. Figure 7 presents some examples where the black vertices are vertex covers:

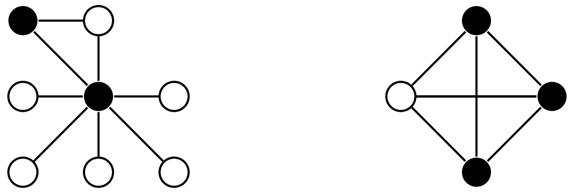

Figure 7: Vertex Covers

Notice that the opposite of a vertex cover (vertices not in the vertex cover) can have no edges in between. Otherwise, this would suggest that there exists an edge with both ends not in the vertex cover, which is a contradiction. That means that the vertices not in the vertex cover is just an independent set! The same logic can also be applied when going from an independent set to a vertex cover. That means that $S \subset V$ is a vertex cover if and only if $\bar{S} = V \setminus S$ is an independent set. This idea will be important for our reduction.

Determining if a vertex cover of a certain size exists in a graph is a very hard problem and is in fact NP-complete. To prove this, let formalize it as a decision problem and define it as the following:

$$\texttt{VERTEX-COVER} = \{\langle G, g\rangle \mid G \text{ has a vertex cover of size } g\}$$

First, we will show that VERTEX-COVER $\in$ NP. Our verifier takes in as input $\langle G, g\rangle$ and $\langle v_1, ..., v_k\rangle$ and checks for each edge in the graph G if one endpoint is in $\{v_1...v_k\}$. If it is, then the a vertex cover exists. This verifier runs in polynomial time since we are simply looping through $v_1...v_k$ for each edge in the graph. Therefore, we can conclude that VERTEX-COVER $\in$ NP.

To prove that VERTEX-COVER is NP-hard, we will reduce from INDEPENDENT-SET to VERTEX-COVER. We will define a reduction $f : \langle G, g\rangle \to \langle G, |V| - g\rangle$ in which V is the vertex set of G and $|V|$ is the total number of vertices.

If $\langle G, g\rangle \in$ INDEPENDENT-SET, there exists a selection of V, say S where $|S| = g$, with no edge between them. Then $\bar{S} = V \setminus S$ is a set of vertices where all edges have to touch. If no edge has both ends in S, every edge has one end in $V \setminus S$. That means that $\bar{S}$ is a vertex cover of size $|V| - g$, so we have shown that $\langle G, g\rangle \in$ INDEPENDENT-SET $\implies \langle G, |V| - g\rangle \in$ VERTEX-COVER.

Similarly, if $\langle G, |V| - g \rangle \in$ `VERTEX-COVER`, then there exists a selection $S \subset V$ where $|S| = |V| - g$ and every edge has at least one end in S. Since S is a vertex cover, $\bar{S} = V - S$ has no edges, so $\bar{S}$ is an independent set of size $|V| - (|V| - g) = g$. Thus, $\langle G, |V| - g \rangle \in$ `VERTEX-COVER` $\implies \langle G, g \rangle \in$ `INDEPENDENT-SET`.

Since we found a valid polynomial reduction, we can conclude that `INDEPENDENT-SET` $\leq_p$ `VERTEX-COVER` and therefore `VERTEX-COVER` is NP-hard.

Since `VERTEX-COVER` $\in$ NP and `VERTEX-COVER` is NP-hard, we conclude that `VERTEX-COVER` is NP-complete.