Vertex Cover

- Vertex cover for a graph $G = (V, E)$ is defined by a subset V' of V such that it contains at least one endpoint of each $e \in E$
- An independent set in $G = (V, E)$ is a subset $V' \subseteq V$ such that for all $u, v \in V'$, the edge $(u, v) \notin E$.
- Decision version of the problem will be: given a positive integer k does G have an independent set with $|V'| \geq k$.
- The relationships between independent sets, cliques and vertex covers are are defined by following result.

Vertex Cover

For a given graph $G = (V, E)$ and a subset $V' \subseteq V$ the following statements are equivalent

(a) V' is a vertex cover for G.

(b) $V - V'$ is an independent set for G.

(b) $V - V'$ is a clique in the complement graph $G^c = (V, E^c)$, where

$$E^c = \{(u, v) | \; u, v \in V \text{ and } (u, v) \notin E\}$$

- So, all the three problems are different versions of each other.
- We will only prove that vertex cover is $\mathcal{NP}$ complete.

Vertex Cover

- To show VC belongs to class $\mathcal{NP}$, just guess a subset and check in p-time whether the subset contains at least one endpoint of every edge and has the appropriate size.
- To show it is $\mathcal{NP}$ hard convert 3-SAT to VC.
- Let an instance of a 3-SAT be defined by

$$\begin{aligned} U &= \{u_1, u_2, \ldots, u_n\}, \text{ and} \\ C &= \{c_1, c_2, \ldots, c_m\} \end{aligned}$$

- We construct a graph $G = (V, E)$ and a positive integer $K \leq |V|$ such that G has a VC of size K or less if and only if C is satisfiable.

Vertex Cover

- The construction consists of two components:
  - `truth setting component`
  - `satisfaction component`
- Also have some additional edges for relating these components.
- For each variable $u_i \in U$ there is a truth setting component $T_i = (V_i, E_i)$, with

$$V_i = \{u_i, \bar{u}_i\}, \text{ and } E_i = \{(u_i, \bar{u}_i)\}.$$

- This means any VC will have either u_i or $\bar{u}_i$ to qualify for a cover.

Vertex Cover

- As there n variables, $|\bigcup_i V_i| = 2n$, and one vertex must be chosen from each V_i.
- For each clause $c_j \in C$, a satisfication component $S_j = (V'_j, E'_j)$, consisting of three vertices and three edges joining them in the form of a triangle:

$$\begin{aligned} V'_j &= \{a_1[j], a_2[j], a_3[j]\} \\ E'_j &= \{(a_1[j], a_2[j]), (a_1[j], a_3[j]), (a_2[j], a_3[j])\} \end{aligned}$$

- Any vertex cover must include at least 2 vertices from V'_j in order to cover the edges in E'_j.
- Giving a total of $2m$ vertices for m clauses.

Vertex Cover

- The only part of construction that depends on which literals occur in which clauses is collection of communication edges.
- How these edges complete construction can be considered best from the view of the requirements that each satisfaction testing component would make its corresponding clause true with the truth values of the literals as enforced by truth setting components.
- For each clause $c_j \in C$, let three literals in c_j be denoted by x_j, y_j and z_j.
- Then communication edges emanating from S_j are given by

$$E''_j = \{(a_1[j], x_j), (a_2[j], y_j), (a_3[j], z_j)\}$$

Vertex Cover

- The construction is complete by setting $K = n + 2m$.
- Note that we are not adding any extra vertices than what already we have.
- This means all communication edges can be covered by the vertices either from $\bigcup_i V_i$ or from $\bigcup_j V_j'$.
- $G = (V, E)$ is now defined as

$$V = \left(\bigcup_{i=1}^{n} V_i\right) \cup \left(\bigcup_{j=1}^{m} V_j'\right)$$

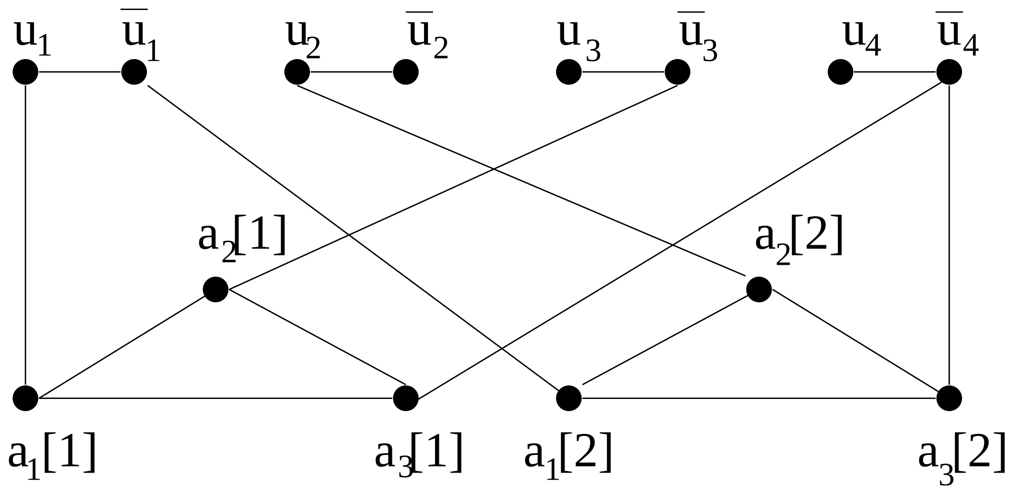
$$E = \left(\bigcup_{i=1}^{n} E_i\right) \cup \left(\bigcup_{j=1}^{m} E_j'\right) \cup \left(\bigcup_{j=1}^{m} E_j''\right)$$

- The construction clearly is p-time.

Vertex Cover

- The example of graph obtained for

$$U = \{u_1, u_2, u_3, u_4\} \text{ and}$$
$$C = \{\{u_1, \bar{u}_3, \bar{u}_4\}, \{\bar{u}_1, u_2, \bar{u}_4\}\}$$



Vertex Cover

- All that has to be proved
  C is satisfiable if and only if G has a vertex cover of size K or less.
- Let V' be a VC for G with $|V'| \leq K$.
- V' must contain at least one vertex from each T_i and at least 2 vertices from each S_j.
- This gives a total of at least $n + 2m = K$ vertices.
- The intersection of V' with truth setting component can be used to obtain a truth assignment $t : U \rightarrow \{\texttt{T},\ \texttt{F}\}$.
- Just set $t(u_i) =$T if $u_i \in V'$ and $t(u_i) =$F if $\bar{u}_i \in V'$.

Vertex Cover

- Consider the three edges in E''_j, only two of these are covered by vertices from $V'_j \cap V'$.
- So one of them must be covered by a vertex from some V_i that belongs to V'.
- But this means corresponding literal either u_i or $\bar{u}_i$ from c_j is true under truth assignment t.
- So c_j is satisfied by t.
- Because it holds for each $c_j \in C$, it follows that t is a satisfying truth assignment for C.

Vertex Cover

- For converse part, let $t : U \rightarrow \{\texttt{T}, \texttt{F}\}$ is a satisfying truth assignment for C.
- The corresponding cover V' includes one vertex each from T_i and two vertices from each S_j.
- The vertex from T_i in V' is u_i if $t(u_i) = \texttt{T}$ and $\bar{u}_i$ if $t(u_i) = \texttt{F}$
- This ensures that at least one of three edges from each set E''_j is covered, because t satisfies each clause c_j.
- Therefore we need to only include in V' the end-points from S_j of other two edges in E''_j, which may or may not be covered by vertices from truth setting components.
- This gives a VC.

Summary of NP Completeness

- Many algorithms have running time which is polynomial in input size. Such algorithms are referred to as p-time algorithms.
- But there are algorithms that need some kind of guided exhaustive search or have to generate huge output.
- These algorithms have exponential running time, and called intractable.
- However, we also have some class of problems for which no algorithms can exist. These are known as undecidable problems.
- Undecidable problems also belong to class of intractable problems.

Summary of NP Completeness

- To study the NP completeness in a formal framework, first we defined abstract problem as a relation from set of problem instances to the set of solutions.
- Then we refined abstract problems to more concrete versions to be able to view both input/output as the strings of symbols.
- The concrete problem is thus obtained by a contraction of solution set to the space of $\{0, 1\}$.
- This definition of concrete problem is thus more naturally referred to as decision version of the problem.
- We also studied how any problem including optimization problems can be reformulated as a decision problem

Summary of NP Completeness

- The main objective behind considering decision problems is that we can conveniently map each problem to that of language recognition problem.
- That is all problems can be studied in the formal language framework.
- The framework consists of alphabets, and a set of strings of alphabets.
- Not all the strings of an alphabet set can be part of the language equivalent to a problem.
- Only the valid strings as the languages corresponding to different decision problems.
- Language encodings of problems are, thus, discussed in details.
- The concept of reasonable encoding was found to be quite important in this framework of formal languages to restrict the encoding from influencing the problem complexity.

Summary of NP Completeness

- Turing machine, which is weakest model for a real computer was introduced.
- Turing machine is capable of just processing a string and producing one symbol as output. Thus it is adequate as a recognizer for a valid strings of a language.
- We also introduced the Nondeterministic Turing Machine.
- Using the concept of NDTM we formalized the notion of the NP class of problem.
- Keeping the convenience of RAM model in view, we defined a similar Nondeterministic RAM model.

Summary of NP Completeness

- Different complexity classes such as: P, $\mathcal{NP}$, CO-$\mathcal{NP}$ are introduced.
- Possible relationship among these classes are discussed.
- We also studied what are known as $\mathcal{NP}$ hard problems.
- Satisfiability problem or the C-SAT is introduced as the basic problem in the class $\mathcal{NP}$ complete.
- An informal proof of C-SAT belonging to $\mathcal{NP}$-complete class was also discussed.

Summary of NP Completeness

- We studied a number of example of polynomial reducibility to prove $\mathcal{NP}$-completeness.
- The problems are:
  - 3-SAT
  - 3-D matching
  - Vertex cover.

