CS 386 – Lecture Notes
Reductions and Approximation: Vertex Cover

Logistics: next week, elections. Project proposals are due October 29.

1 Review: polynomial-time reductions

Definition 1. *For decision problems A and B, we write $A \leq_p B$ if there is a polynomial-time computable function f such that, for every input x,*

$$x \in A \quad \Longleftrightarrow \quad f(x) \in B.$$

Thus, an algorithm for B can solve A: transform the input using f, then run the algorithm for B. In particular, if $A \leq_p B$ and $B \in \mathrm{P}$, then $A \in \mathrm{P}$.

Definition 2. *A problem B is **NP-hard** if $A \leq_p B$ for every $A \in \mathrm{NP}$. It is **NP-complete** if it is both NP-hard and in NP.*

If any problem in NP has no polynomial-time algorithm, then no NP-complete problem has a polynomial-time algorithm. Otherwise, reducing that problem to an NP-complete problem would give a polynomial-time algorithm for it.

2 Vertex Cover

Definition 3. *A **vertex cover** of an undirected graph $G = (V, E)$ is a set $S \subseteq V$ such that every edge has at least one endpoint in S. The decision problem is*

$$\mathrm{VC} = \{\langle G, k\rangle : \textit{there exists a vertex cover } S \textit{ with } |S| \leq k\}.$$

[Scribe's note: The handwritten definition uses $|S| < k$; the standard convention, used in the reduction below, is $|S| \leq k$.]

3 Reducing 3SAT to Vertex Cover

Theorem 1. $\mathrm{3SAT} \leq_p \mathrm{VC}$. *Consequently, Vertex Cover is NP-complete.*

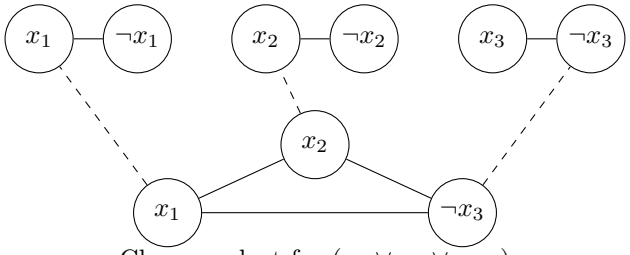
Let φ be a 3-CNF formula with n variables and m clauses. We construct a graph G and set $k = n + 2m$.

1. **Variable gadgets.** For each variable x_i, create two vertices labeled x_i and $\neg x_i$, joined by an edge.
2. **Clause gadgets.** For each clause, create three distinct vertices, one for each literal occurrence, and join them in a triangle.
3. **Connecting edges.** Join each clause vertex to the variable-gadget vertex with the *same literal* label.

Example. The formula in the notes is

$$\varphi = (x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (x_1 \vee x_2 \vee \neg x_3).$$

Here $n = m = 3$, so $k = 9$. The following illustrates one clause gadget and its connections; the construction repeats this for every clause.



Clause gadget for $(x_1 \vee x_2 \vee \neg x_3)$

Polynomial running time. The graph has $2n + 3m$ vertices and $n + 6m$ edges. It can be constructed in polynomial time.

Forward direction: a satisfying assignment gives a cover

Suppose φ has a satisfying assignment. Construct S as follows:

- From each variable gadget, select the vertex whose literal is true.
- In each clause triangle, leave one true literal occurrence unselected and select the other two vertices.

Every clause has a true literal, so this is possible, and $|S| = n + 2m = k$. To check that S is a vertex cover, consider all three types of edges:

1. Each variable edge is covered by its selected endpoint.
2. Each triangle edge is covered because two vertices of the triangle are selected.
3. Each connecting edge is covered at its clause endpoint unless that endpoint was left unselected. In that case its literal is true, so its variable endpoint is selected.

Hence $\langle G, k \rangle \in \text{VC}$.

Reverse direction: a cover gives a satisfying assignment

Suppose G has a vertex cover S with $|S| \leq n + 2m$. Every variable edge requires at least one selected endpoint, and every clause triangle requires at least two selected vertices. These gadgets have disjoint vertex sets, so $|S| \geq n + 2m$. Thus equality holds: exactly one vertex is selected per variable gadget and exactly two per clause triangle.

Set x_i to true exactly when the variable vertex x_i belongs to S. Each triangle has one unselected vertex. Its connecting edge must be covered by the corresponding variable vertex, so that literal is true. Every clause therefore has a true literal, and φ is satisfiable.

Finally, VC is in NP: given S, we can check its size and whether it covers every edge in polynomial time. Together with the reduction from 3SAT, this proves NP-completeness. □

4 A linear-programming approximation for Vertex Cover

The optimization version asks for a vertex cover of minimum size. Introduce a variable for each vertex:

$$x_v = \begin{cases} 1, & \text{if we select } v, \\ 0, & \text{otherwise.} \end{cases}$$

An exact integer-programming formulation is

$$\begin{aligned} \text{minimize} \quad & \sum_{v \in V} x_v \\ \text{subject to} \quad & x_u + x_v \geq 1 && \text{for every } \{u, v\} \in E, \\ & x_v \in \{0, 1\} && \text{for every } v \in V. \end{aligned}$$

The edge constraints say that at least one endpoint of every edge must be selected. For the decision version, we additionally ask whether $\sum_{v \in V} x_v \leq k$ is possible.

Relaxation and rounding

Replace $x_v \in \{0, 1\}$ by $0 \leq x_v \leq 1$. This is a **linear-programming relaxation**, which can be solved in polynomial time. Let x^* be an optimal fractional solution. Round it by setting

$$\bar{x}_v = \begin{cases} 1, & x_v^* \geq \frac{1}{2}, \\ 0, & x_v^* < \frac{1}{2}. \end{cases} \qquad S = \{v \in V : \bar{x}_v = 1\}.$$

[Scribe's note: The threshold must include equality: round up when $x_v^* \geq \frac{1}{2}$. Using a strict inequality could leave an edge uncovered when both endpoints have value $\frac{1}{2}$.]

Theorem 2. *The rounding algorithm returns a vertex cover of size at most twice the minimum possible size.*

Proof. First, S is a vertex cover. For each edge $\{u, v\}$, feasibility gives $x_u^* + x_v^* \geq 1$. At least one endpoint therefore has value at least $\frac{1}{2}$, and that endpoint is selected by rounding.

Next, for every vertex v,

$$\bar{x}_v \leq 2x_v^*.$$

Indeed, if $\bar{x}_v = 0$, the inequality follows from $x_v^* \geq 0$; if $\bar{x}_v = 1$, it follows from $x_v^* \geq \frac{1}{2}$. Summing gives

$$|S| = \sum_{v \in V} \bar{x}_v \leq 2 \sum_{v \in V} x_v^* \leq 2\,\mathrm{OPT},$$

where OPT is the minimum size of an integral vertex cover. The final inequality holds because every integral cover is feasible for the relaxation, so the fractional optimum is no larger than OPT. □

In the notation of the handwritten notes, if the fractional solution satisfies $\sum_v x_v^* \leq k$, rounding produces a cover with at most $2k$ vertices. Thus this is a polynomial-time **2-approximation algorithm**.