1 Vertex Cover Problem

Consider the following problem.

- You are a manager for Acme Security Guards, Inc.
- You are assigned to guard a building whose floorplan is a network of hallways:
  - Each hallway is a straight line; hallways meet at hubs.
  - You must station guards at a subset of the hubs.
  - A guard stationed at a hub can watch all hallways that meet at that hub.
  - Goal is to assign as few guards as possible to the building, without leaving any hallway unguarded.

Let’s abstract this problem.

- Let $G = (V, E)$ be an undirected graph.
- A vertex cover of $G$ is a subset of vertices $V' \subseteq V$ such that, for each $(u, v) \in E$, at least one of $u$ or $v$ is in $V'$.

**Problem:** given a graph $G$, find its smallest vertex cover.

How hard is this problem?

2 Vertex Cover Is Hard

Will show that vertex cover is an NP optimization problem.

- Decision problem (VERTEX-COVER): given $G$, does it admit a vertex cover of size at most $k$?
• **Lemma:** VERTEX-COVER is NP-complete.

• **Pf:** first, show that it is in NP.

• The cover itself is the certificate; details left as exercise.

• To show NP-hardness, will reduce from INDEPENDENT-SET.

• Let \((G, j)\) be an input to INDEPENDENT-SET, and let \(G\) contain \(n\) vertices.

• Corresponding instance of VERTEX-COVER is \((G, n - j)\).

• **Claim 1:** if \(G = (V, E)\) has an independent set of size at least \(j\), then it has a vertex cover of size at most \(n - j\).

• **Pf:** let \(S\) be an independent set of size \(j\) in \(G\).

• Consider the set of \(n - j\) vertices \(T = V - S\).

• We claim that \(T\) is a vertex cover for \(G\).

• Indeed, no edge of \(G\) connects two vertices of \(S\), since it is an independent set!

• Hence, every edge in \(G\) has at least one endpoint in \(T\) and so is covered by \(T\).

• **Claim 2:** if \(G\) has a vertex cover of size at most \(k\), then then it has an independent set of size at least \(n - k\).

• **Pf:** let \(T\) be a vertex cover of size \(k\) for \(G\).

• Again, let \(S = V - T\). \(|S| = n - k\).

• No two vertices of \(S\) are connected; otherwise, the resulting edge would not be covered by \(T\).

• Hence, \(S\) forms an independent set in \(G\). QED

### 3 OK, Now What?

NP-hardness does not mean you can give up!!!

• What do you do if you want to solve an NP optimization problem?

• **Idea 1:** use an exponential-time algorithm.

• If your inputs are all small, this is fine. *(Example: the old GRE analytic section)*

• (Also, if you never hit the worst case in practice, this is fine!)

• However, this approach does not scale!

• **Idea 2:** add constraints until the problem becomes polynomial-time solvable.

• **Example:** if \(G\) is bipartite, the size of a minimum vertex cover equals the size of a maximum matching (Koenig’s Theorem), which can be found in polynomial time.
• But your problem may not be constrainable!

• **Idea 3**: find a polynomial-time optimization algorithm whose answer is “close” to best possible.

• Heuristic algorithms for optimization (gradient descent, simulated annealing, Gibbs sampling, etc etc etc) try various hacks to get close to either local or global optimum.

• Can we quantify how close a heuristic gets to the optimum?

• Maybe empirically, if we can afford to compute true optimum for enough inputs to run a benchmark.

• Sometimes, however, we can prove that a heuristic always gets close to the optimum!

Time for some definitions.

• Let $P$ be an optimization problem whose goal is to find a feasible solution of minimum cost.

• For an arbitrary input $x$ to $P$, let $C^*(x)$ be the cost of an optimal solution to $x$.

• Let $A$ be an algorithm that computes a feasible solution to $P$, and let $C(x)$ be the cost of its solution to input $x$.

• **Defn**: we say that $A$ is an $f(n)$-*approximation algorithm* for $P$ if, for any input $x$ of size $n$, we have

$$\frac{C(x)}{C^*(x)} \leq f(n).$$

• In other words, for an input of size $n$, $A$ gives a solution no worse than $f(n)$ times the optimum.

• $f(n)$ is called the *approximation ratio* for $A$.

• A similar definition holds for maximization problems.

Approximation algorithms are heuristics that come with a guarantee: they never return a solution more than $f(n)$ times worse than the optimum.

### 4 Approximating Vertex Cover

We will give polytime algorithm for vertex cover whose solution is never worse than twice the optimum.

• Given graph $G$, construct a cover $T$ as follows.

• Pick an arbitrary edge $e = (u, v)$ in $G$, and add both $u$ and $v$ to $T$.

• Delete $u$, $v$, and their incident edges from $G$.

• Repeat until $G$ contains no more edges.
• Call this algorithm FAST-COVER.

First, does FAST-COVER produce a correct vertex cover?

• No edge is removed from $G$ unless at least one of its endpoints is covered by a vertex in $T$.
• When algo terminates, all edges have been removed.
• Hence, $T$ is a vertex cover for $G$. QED

Second, does FAST-COVER run in polynomial time?

• Consider adjacency list representation of $G$.
• If $G$ has $n$ vertices, it takes time $O(n)$ to find an edge and $O(n)$ to remove it and all adjacent edges from $G$.
• Hence, algorithm is surely $O(n^2)$.
• (We can perhaps do better, but right now, just see that it’s in P.)

Finally, is FAST-COVER an approximation algorithm?

• **Lemma:** FAST-COVER is a 2-approximation algorithm for vertex cover.
• How are we going to prove this?
• Proof will follow a sequence of steps:
• Let $x$ be an arbitrary instance of vertex cover.
• First, find a lower bound $L(x)$ on the cost $C^*(x)$ of a minimal cover for $x$.
• Then, find an upper bound $U(x)$ on the cost $C(x)$ of FAST-COVER’s solution, in terms of $L(x)$.
• Conclude that the approximation ratio for FAST-COVER is bounded as follows:

$$\frac{C(x)}{C^*(x)} \leq \frac{C(x)}{L(x)} \leq \frac{U(x)}{L(x)}.$$

• Finally, show that $\frac{U(x)}{L(x)} \leq 2$.

### 5 Proving The 2-Approximation

Let $G = (V, E)$ be an input to vertex cover.

• First, we lower-bound optimum.
• Let $M \subseteq E$ be the list of edges of $G$ whose endpoints are chosen by FAST-COVER.
• By construction, no two edges in $M$ share an endpoint (that is, $M$ is a matching for $G$.)
• Hence, any valid vertex cover for $G$ must contain at least one endpoint of every edge in $M$.

• Hence, every valid vertex cover for $G$ has size at least $|M|$.

One step down, two to go.

• Second, we upper-bound the algorithm’s solution.

• FAST-COVER adds both endpoints of every edge in $M$ to the cover.

• Hence, its solution has size $2|M|$.

Finally, put upper and lower bounds together.

• We have shown that $C(G) = 2|M|$, while $C^*(G) \geq |M|$.

• Hence, FAST-COVER’s approximation ratio for $G$ is

\[ \frac{C(G)}{C^*(G)} \leq \frac{2|M|}{|M|} = 2. \]

• Conclude that FAST-COVER is a 2-approximation algorithm for vertex cover. QED

6 Best Known Results

• Above approximation result is trivially tight (find examples!).

• Best known approximation algorithm for general problem gives ratio of 2 in limit of large $G$, but saves a little bit (factor of $\log \log |V|/\log |V|$) for small graphs (Halperin 2000).

• Equally good approximations known for version in which vertices have non-negative weights and the goal is to minimize the weight of the cover.

• Known not to be approximable to within 1.1666 unless P = NP. (Hastad 1997).

• If Unique Games Conjecture is true, not approximable to within a constant factor better than 2 unless P = NP. (Knot and Regev 2003).