Algorithmic Modeling and Complexity **Fall 2003**

Lecture 4: September 11, 2003

Lecturer: J. van Leeuwen *Scribe: B. de Boer*

4.1 Overview

This lecture introduced Fixed Parameter Tractable (FPT) problems. An example of an FPT problem is the Vertex Cover problem. The Vertex Cover problem can be seen in two ways: as an *optimization* problem and as a *decision* problem. Both these problems were discussed. Later on, the lecture also dealt with approximating the Vertex Cover problem and computing a *connected* vertex cover.

4.2 The Vertex Cover Problem

Let $G =< V, E >$ be an unweighted graph.

Definition 4.1 *A subset $S \subseteq V$ is a vertex cover if every edge of G is incident to a node of S, meaning that for every edge in G at least one end-point of the edge is in S.*

a vertex cover 'covers' all edges of the network. You could look at this in the following way: Placing a guard in every node of the vertex cover guarantees that all edges are guarded. There is another reason why vertex covers are important as well:

Proposition 4.2 *S is a vertex cover if and only if $V - S$(the complement of S) is an independent set.*

Proof: ($\Rightarrow$) Suppose S is a vertex cover. Consider any two nodes u, $v \in V - S$. If there would be an edge between u and v, then (because S is a vertex cover) u or v must be in S. Contradiction! Thus there can be no edge between u and v and $V - S$ is an independent set.

($\Leftarrow$) Suppose $V - S$ is an independent set. Consider any edge (u, v). Since $V - S$ is an independent set, u and v cannot both be elements of $V - S$. Thus at least one of the nodes u and v must be an element of S. Thus S is a vertex cover. ■

4.2.1 Optimization version and decision version

Given their interpretation, our objective will be to find vertex covers of small size. One can distinguish between two types of problems.

- **optimization problem**: compute a minimum size vertex cover.
- **decision problem**: given k, decide whether the network has a vertex cover of size $\leq k$. (And perhaps: construct one if the answer is yes.)

In many cases the optimization and decision versions of a problem are related in an efficient manner. We show this for vertex cover.

Lemma 4.3 *The optimization version of the Vertex Cover problem is solvable in polynomial time if and only if the decision version of the Vertex Cover problem is solvable/decidable in polynomial time.*

Proof: ($\Rightarrow$) Proof by computing the (size of the) minimum vertex cover.

($\Leftarrow$) By searching down from $k = n$ and calling on the decision algorithm, we can determine in at most n steps the *size* k of the minimum vertex cover of G. We can compute a concrete minimum size vertex cover recursively as follows starting with an initially empty set.

Assume $E \neq \emptyset$. Pick any edge $(u, v) \in E$. We know that *at least one* of the nodes u and v must belong to the vertex cover. Determine two subgraphs:

G_u: G with all edges incident to u removed,
G_v: G with all edges incident to v removed.

and remove all isolated nodes from both networks.

Determine the size of the minimum vertex cover of G_u and G_v by means of the decision algorithm. Either G_u or G_v (or both!) has a minimum size vertex cover of size $k - 1$.

If G_u has a minimum size vertex cover of size $k - 1$, then add $\{u\}$ to the cover and recurse on G_u. If G_u does not have a minimum size vertex cover of size $k - 1$, then add $\{v\}$ to the cover and recurse on G_v.

After k recursions a size-k, i.e. minimum vertex cover will be found. Every recursive step involves $\leq n$ calls of the decision procedure. ■

4.3 Computing minimum vertex covers exactly

Let $G =< V, E >$ be a network, OPT the minimum size of any vertex cover of G. How to compute a minimum size vertex cover:

We give a branching algorithm that operates on instances (G, S) with G the original network and $S \subseteq V$ initially empty. We want to break down G and accumulate nodes of a (possibly minimum) vertex cover in S.

In a *branching algorithm* a problem instance is repeatedly split into smaller problem instances, leading to a tree of subproblems in which only the problems at the leaves matter. Typically, we are after the best solution in any leaf (unless we decide to branch further).

This branching algorithm will repeatedly pick leaves and carry out the following rule:

> *Branching rule.* Given (H, S) and $H \neq \emptyset$. Pick an edge $(u, v) \in H$. Split the problem into two subproblems:
>
> - $(H_u, S \cup \{u\})$ with H_u equal to H with all edges incident to u removed and isolated nodes thrown away.
> - $(H_v, S \cup \{v\})$ with H_v equal to H with all edges incident to v removed and isolated nodes thrown away.

Develop the branching tree in breadth-first manner.

Lemma 4.4 *The first level k of the branching tree that contains a node with label $(\emptyset, S)$ gives the size of the minimum vertex cover and S is such a minimum vertex cover.*

Proof: Observe that the algorithm maintains the following invariant:

(a) for at least one leaf (H, S): "$S\cup$ a minimum vertex cover for H" is a minimum vertex cover for G.

The proof of the invariant follows by inspecting the branching rule again. If $(u, v) \in H$, then at least one of the nodes u and v will belong to a minimum vertex cover of H. Clearly "$\{u\}\cup$ a minimum vertex cover of H_u" and "$\{v\}\cup$ a minimum vertex cover of H_v" *both* are vertex covers of H but only at least one of them will also be a minimum vertex cover of H. The proof of the invariant property is now easily completed.

Consider the algorithm as it grows the branching tree level after level. After completing the k'th level, all nodes (H, S) in the level will have $|S| = k$.

If level k contains a node that cannot be split further, i.e. a node $(\emptyset, S)$, then this node must be a leaf corresponding to property *(a)*. The lemma follows. ∎

Theorem 4.5 *Minimum vertex covers can be computed in $O(2^{OPT} \cdot |G|)$ time.*

Proof: The branching algorithm will continue precisely to level OPT. This involves branching in $1 + 2 + 4 + \ldots + 2^{OPT-1} = 2^{OPT} - 1$ nodes and each call of the branching rule takes $O(|G|)$ time. ∎

Exercise. Show how to implement the branching algorithm efficiently using a queue.

4.3.1 Fixed parameter tractable problems

A problem 'with a parameter k' is called *fixed parameter tractable* (FPT) if it can be solved or decided by an algorithm within a running time $O(f(k).poly(n))$, for some function f. For fixed k, this is polynomial time. The class of fixed parameter tractable problems is denoted as: FPT [3].

Corollary 4.6 *Vertex cover problem $\in$ FPT.*

Proof: Do the branching algorithm for k levels and see if any node labelled $(\emptyset, S)$ is encountered. If so, the answer is yes. If not, the answer is no. Running time: $O(2^k \cdot |G|) = O(2^k \cdot (n+m)$. ∎

The decision version of the Vertex Cover problem can be decided within better bounds than we derived. The following table shows very recent progress.

year	author(s)	upperbound
1993	Buss	$k \cdot n + 2^k \cdot k^{2k+2}$
1998	Balasubramanian, Fellows and Raman	$k \cdot n + 1.324718^k \cdot k^2$
1999	Downey, Fellows and Stege	$k \cdot n + 1.31951^k \cdot k^2$
1999	Niedermeier and Rossmanith	$k \cdot n + 1.29175^k \cdot k^2$
1999	Stege and Fellows	$k \cdot n + \max\{1.25542^k \cdot k^2, 1.2906^k \cdot k\}$
2000	Niedermeier and Rossmanith	$k \cdot n + 1.2906^k$
2001	Chen, Kanj and Jia	$k \cdot n + 1.2852^k$

The Dominating Set problem is *not* known to be $\in$ FPT, but the dominating set problem restricted to planar networks *is* in FPT.

Theorem 4.7 (Downey and Fellows, 1995) *The Dominating Set problem is solvable in $O(11^k * (n+m))$ time.*

4.4 How to approximate minimum vertex covers fast

There is a variety of algorithms for computing 'approximately minimum' vertex covers efficiently but the best result is still the following.

Theorem 4.8 *The Vertex Cover problem can be approximated in polynomial time within the performance ratio of 2 (thus leading to vertex covers of size at most $2 \cdot OPT$).*

```
Algorithm G1
$S := \emptyset$
As long as there are edges left do:
    pick an edge $(u, v)$
    $S := S \cup \{u, v\}$
    delete nodes $u$ and $v$ and all edges incident to them (and remove all isolated nodes)
return($S$)
```

Lemma 4.9 *Algorithm G1 computes a vertex cover of size $\leq 2 \cdot OPT$.*

Proof: *(a)* S is a vertex cover. Edges can be removed only if (at least) one of their end points has been put in S.

(b) $|S| \leq 2 \cdot OPT$. When we choose an edge, the optimum would have to include at least one node. The algorithm includes two. ■

Observation: The edges (u, v) picked by the algorithm form a *matching* (i.e. a set of edges $M \subseteq E$ such that no two edges of M are incident.)

Exercise. Argue that the edges picked form a maximal matching.

Another algorithm, due to Gavril ($\leq$ 1979).

> **Algorithm** G2
>
> Compute any maximal or even maximum matching M in G.
>
> Take the end points of the edges in M as vertex cover.

Lemma 4.10 *Algorithm G2 gives a vertex cover of size $\leq 2 \cdot OPT$.*

Proof: The algorithm returns a vertex cover, because an edge (u, v) is either:

- part of the matching and thus has both its nodes in the vertex cover, or
- has one endpoint that is in the vertex cover, or
- has no endpoints that are in the vertex cover, but this isn't possible, because then (u, v) could be added to the matching, contradicting that it was maximal.

Note that any edge $\in M$ must contain at least one node of the optimum cover. Thus

$$|M| \leq OPT \leq |\text{vertex cover}| = 2 \cdot |M| \leq 2 \cdot OPT$$

Thus, $|\text{vertex cover}| \leq 2 \cdot OPT$. ■

Theorem 4.11 (Bar-Yehuda, Even, 1985) *There exists a polynomial-time approximation algorithm for the Vertex Cover problem with performance ratio $2 - \frac{loglogn}{logn}$.*

Håstad (1997) proved that the Vertex Cover problem cannot be approximated in polynomial time within a performance ratio ≤ 1.1666 unless some big open problems in complexity theory are solved.

4.5 The Connected Vertex Cover problem

Assume without loss of generality that G is connected. It turns out that a stronger result can be obtained for the vertex cover problem. Let OPT again denote the minimum size of *any* vertex cover in G.

Theorem 4.12 (Savage, 1982) *In polynomial time, one can compute a connected vertex cover of size* $\leq 2 \cdot OPT$.

The algorithm turns out to be very simple, but it will be harder to actually show that it does the job.

```
Algorithm S

Compute a depth-first search tree $T$ of $G$.

Determine $I(T)$ = the set of internal nodes of $T$.

Return $I(T)$
```

$I(T)$ is obviously connected. We now show that it is a vertex cover and that it must have size $\leq 2 \cdot OPT$.

Lemma 4.13 $I(T)$ *is a vertex cover.*

Proof: Any edge is either a tree edge or a back-edge. In both cases, it is incident to at least one internal node. ∎

Exercise. Show that the leaves of depth-first search tree form an independent set.

Lemma 4.14 $|I(T)| \leq 2 * OPT$.

Proof:

Consider T. Then $|I(T)| = \#nodes - \#leaves$.

It is easy to show by induction that $\#leaves = 1 + \sum_{v \in I(T)} (sons(v) - 1)$, where $sons(v)$ is the number of sons of v in T. In order to estimate this expression we use a maximum matching M of the depth-first search tree T. We show that M can be changed into a maximum matching with a special property.

Let r be the root of T.

Definition 4.15 *A node w is called* free *when it is not matched in M.*

Definition 4.16 *A free node w with $w \neq r$ is called free$^+$ if the father v of w has at least one other son, say u, and moreover $(u, v) \in M$.*

Claim 4.17 *Every tree T has a maximum matching M with the property that every free node $\neq r$ is free$^+$.*

Proof: Let M be a maximum matching in T. Suppose M does not have the stated property. We show that M can be transformed into a maximum matching that does.

Let a lowest free node w that is not free$^+$ occur at level k in T. Assume w.l.o.g. that $k > 0$, i.e. level k is not the root level. We show that we can modify M and establish the property in level k, by at best creating more free nodes w that are not free$^+$ in levels $< k$. For any free node w in level k that is not free$^+$, there can be the following cases. In all cases, let $v = father(w)$.

Case 1: deg$(v) = 1$. If v were free, then we could add (w, v) to M, contradicting that M is maximum. Thus v is not free, which implies that it has a father z and that $(v, z) \in M$. Now delete (v, z) from M but add (w, v) to it. We maintain a maximum matching, but w is no longer free in it.

Case 2: deg$(v) \geq 2$ but for no son u of v we have $(v, u) \in M$. As in *case 1* one argues that v is not free. Because w is assumed not to be free$^+$, this implies that v must have a father z and is matched by the edge $(v, z) \in M$. Again delete (v, z) from M but add (w, v) to it: because no sons of v were matched this switch keeps a matching. We maintain a maximum matching, but w is no longer free in it. All other free sons of v have become free$^+$.

By modifying M inductively from level k upward towards the root, we 'push out' all free nodes that aren't free$^+$. ■

Thus assume, as we may, that the chosen maximum matching M of T has the property that every free node w with $w \neq r$ is free$^+$. Let F be the set of free nodes.

$$\#leaves = 1 + \sum_{v \in I(T)} (sons(v) - 1) \geq 1 + \sum_{i=1}^{k} (sons(v_i) - 1),$$

where $v_1, \ldots, v_k$ are the nodes $\in I(T)$ that have at least one son that is free. As free sons are free$^+$, it means that every v_i also has at least one non-free son. Thus $\sum_{v_1}^{v_k}(sons(v) - 1)$ is greater than or equal to the collective number of *free sons* of the v_i's and there are at least $|F| - 1$ of these. (Observe that the root might be in F and it has to be excluded in the count.) Hence:

$$\#leaves \geq 1 + \sum_{v_1}^{v_k} (sons(v) - 1) \geq 1 + (|F| - 1) = |F|$$

where $|F| = n - 2|M|$.

Consequently $|I(T)| = n - \#leaves \leq n - |F| = 2|M|$.

Letting $M(G)$ be any maximum matching of G we can conclude :

$$|M| \leq |M(G)| \leq OPT \leq |I(T)| \leq 2 \cdot |M|,$$

which proves that $|I(T)| \leq 2 \cdot OPT$. ■

References

[1] R. Bar-Yehuda, S. Even. A local-ratio theorem for approximating the weighted vertex cover problem. *Annals of Discr. Mathematics* 25 (1985) 27-44.

[2] J. Chen, I.A. Kanj, and W. Jia. Vertex cover: Further observations and further improvements. *J. of Algorithms* 41 (2001) 280-301.

[3] R.G. Downey, M.R. Fellows. *Parameterized complexity.* Springer-Verlag, New York, 1999.

[4] J. Håstad, Some optimal inapproximability results. *Proc. 29th Ann. ACM Symposium on Theory of Computing*, 1997, pp 1-10.

[5] C. Savage. Depth-first search and the vertex cover problem. *Information Processing Letters* 14 (1982) 233-235.