Fixed-Parameter Tractable Algorithms - Vertex Cover

Anil Maheshwari

anil@scs.carleton.ca
School of Computer Science
Carleton University
Canada

Outline

Problem Statement

Vertex Cover in Graphs

Input: A simple undirected graph $G = (V, E)$.

Output: A subset $S \subseteq V$ of smallest cardinality such that for each edge $e = (u, v) \in E$, at least one of u or v is in S.

Complexity Results

Let G be a simple undirected graph, and let k be the cardinality of its minimum vertex cover.

1. **NP**-Complete for graphs.
2. Polynomial Time Approximation Algorithm.
3. Exact (FPT) Algorithms:
    3.1 A naive algorithm running in $O(|V|^{k+1})$ time.
    3.2 An algorithm running in $O(|V|2^k)$ time.
    3.3 An algorithm running in $O(|V| + |E| + k^2 2^k)$ time.
    3.4 $\cdots$

Note: FPT algorithms are polynomial in graph parameters $|V|$ and $|E|$, but exponential in k - the size of the vertex cover. If k is small, these algorithms are efficient.

A Simple Approximation Algorithm

Approximation via Maximal Matching

A *matching* $M \subseteq E$ in $G = (V, E)$ is a collection of edges so that no two edges in M are incident to the same vertex.

Matching M is *maximal*, if every other edge in $E \setminus M$ shares an end point with some edge in M.

Approx Vertex Cover Algorithm:

1. Compute a maximal matching M of G.
2. Let $S \subseteq V$ be the set of vertices incident on the edges in M.
3. Return S as an approximation to vertex cover of G.

Observation

Any optimal vertex cover $S^* \subseteq V$ of G satisfies $|S^*| \geq |M|$.

Approximation via Maximal Matching (contd.)

Approx Vertex Cover Algorithm:

1. Compute a maximal matching M of G.
2. Let $S \subseteq V$ be the set of vertices incident on the edges in M.
3. Return S as an approximation to vertex cover of G.

Observation

Set of vertices in S forms a vertex cover of G. Moreover, the graph $G \setminus S$ is an independent set.

Claim

$|S| = 2|M| \leq 2|S^*|$. Thus, the above algorithm is a 2-approximation algorithm for the vertex cover problem. The algorithm runs in $O(|V| + |E|)$ time.

FPT Algorithms

A Brute-Force Algorithm

Problem: Whether $G = (V, E)$ has a vertex cover of size $\leq k$?

Easy solution:

- Consider all subsets $S \subseteq V$ of size k.
- Check whether $G \setminus S$ is an independent set.

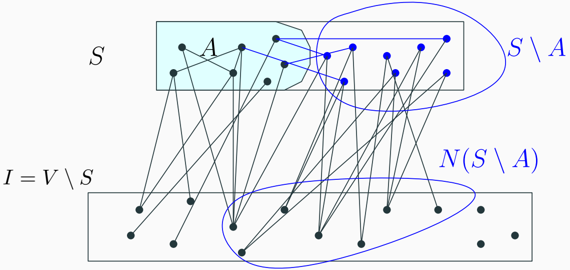
Time Complexity: $\binom{n}{k} O(n + m) = O(n^k(n + m))$, where $n = |V|$ and $m = |E|$.

A Brute-Force Algorithm (contd.)

Problem: Whether $G = (V, E)$ has a vertex cover of size $\leq k$?

1. Find a Maximal Matching M of G.
2. $|M| > k$, return G has no vertex cover of size $\leq k$.
3. Let S be the set of vertices constituting the edges in M.
   Note: S forms a vertex cover of G and $I = V \setminus S$ is an independent set.
4. Consider all possible subsets A of S of size $\leq k$ and check whether $A \cup (N(S \setminus A) \cap I)$ is a vertex cover of G of size at most k. If true, output $A \cup (N(S \setminus A) \cap I)$ as the vertex cover. ($N(X)$ represents neighbors of vertices in X in G.)

A Brute-Force Algorithm (contd.)



S is a vertex cover and I an independent set of G.

Observation

Let S be a vertex cover of $G = (V, E)$. For a subset $A \subseteq S$, $A \cup (N(S \setminus A) \cap I)$ is a vertex cover of G if and only if there are no edges in E such that both of its end points are in $S \setminus A$.

Analysis of Brute-Force Algorithm (contd.)

1. Finding maximal matching M in G requires $O(n+m)$ time.
2. Number of all possible subsets of size at most k of S is $2^{2k} = 4^k$.
3. Checking whether the set $A \cup (N(S \setminus A) \cap I)$ forms the vertex cover of size at most k requires $O(n+m)$ time.
4. Thus, the overall complexity is $4^k n^{O(1)}$.
5. The time complexity is of type $f(k)n^{O(1)}$ - a function in k (may be exponential) and a polynomial function in the size of G.

Fixed-Parameter Tractable

A problem is said to be *fixed parameter tractable* with respect to a parameter k if there is an algorithm with running time $f(k)n^{O(1)}$, where n is the size of the problem and f is independent of n.

Example: Vertex Cover is FPT.

A Branch-and-Bound Algorithm

Observation

For each edge $e = (uv)$ of G, any vertex-cover of G contains at least one of u or v.

Proof: Follows from definition of vertex cover of G.

A FPT Algorithm for Vertex Cover

Algorithm VertexCoverFPT(G, k)

1. if G has no edges then return TRUE
2. if $k = 0$ then return FALSE
3. Let $e = (uv) \in E$ be an edge of G
4. if VertexCoverFPT$(G - u, k - 1)$ then return TRUE
5. if VertexCoverFPT$(G - v, k - 1)$ then return TRUE
6. return FALSE

Note: Above algorithm is a Decision Algorithm - answers whether G has a vertex cover of size $\leq k$?

With some extra work, we can also find the vertex cover $S \subseteq V$ of size $\leq k$.

Correctness

The correctness of the algorithm is based on induction on the size of the vertex cover k.

- If $k = 0 \implies G$ has no edges and Step 1 returns TRUE.

- Let $G = (V, E)$ be a graph with vertex cover S of size $k > 0$.

- To cover the edge $e = (uv)$, S must contain at least one of u or v.

- If $u \in S$, the graph $G - u$ (i.e., remove u and all its incident edges) has vertex cover of size at most $k - 1$. Step 4 returns TRUE.

- If $u \notin S$, then $v \in S$, $G - v$ has vertex cover of size at most $k - 1$. Step 5 returns TRUE.

- If both returns FALSE, then clearly G doesn't have a vertex cover of size $\leq k$.

Complexity Analysis

Observe that

- Recursion 'tree' is a complete binary tree of height k.

- It consists of 2^k leaves and 2^{k-1} internal nodes.

- Each internal node requires computation time of $O(|V|)$ (e.g. using adjacency list representation of graphs).

- For each leaf node, we need to check whether there are no edges in the remaining graph.

- Overall Running Time = $(2^k + 2^{k-1})O(|V| + |E|) = O((|V| + |E|)2^k)$

Result 1

Let $G = (V, E)$ be a simple undirected graph that has a vertex cover of size at most k. We can find the minimum vertex cover of G in $O((|V| + |E|) \times 2^k)$ time.

Kernelization

Kernel

- $\langle Q, k \rangle \xrightarrow{\mathcal{A}} \langle Q', k' \rangle$
  Given a problem instance Q with parameter k, we will execute an algorithm $\mathcal{A}$, running in polynomial time, to obtain an equivalent instance Q' such that Q has a solution if and only if Q' has a solution.
- We say $\mathcal{A}$ is a *kernelization algorithm* if the size of Q' and k' can be bounded by some function of k (and independent of the size of problem Q.) It will be ideal to bound the size of Q' and k' by a polynomial function in k, preferably linear or quadratic functions.
- The kernelization algorithm $\mathcal{A}$ is usually broken down as a set of rules. For example, for the vertex cover problem, a simple rule is to remove all vertices of degree 0, and the resulting graph has a vertex of size $\leq k$ if and only if the original graph has a vertex cover of size $\leq k$.

Observations

Observation (high-degree vertices)

If G has a vertex u of degree $> k$. Let $S \subseteq V$ be a vertex-cover of G with $|S| \leq k$. Then $u \in S$.

Proof: If $u \notin S$, then all its neighbours must be in S. But u's has $> k$ neighbors and $|S| \leq k$.

$\square$

$\implies$ We can place u in the vertex cover and remove u and all its incident edges in G, and seek for a vertex cover of size at most $k - 1$ in the resulting graph.

Observation

Observation

Let S' be the set of all vertices in G whose degree is $> k$. Let G' be the graph obtained from G by removing all vertices in S' (and their incident edges). G has a vertex cover of size $\leq k$, if and only if, G' has a vertex cover of size $\leq k' = k - |S'|$.

Observation (contd.)

Observation

Let S' be set of all vertices in G whose degree is $> k$. Let G' be the graph obtained from G by removing all vertices in S' (and their incident edges). The degree of each vertex in G' is $\leq k$.

Observation (contd.)

Observation

If graph G' has more than kk' edges, then G' doesn't have any vertex cover of size $\leq k'$.

Proof: Each vertex in the cover of G' can cover at most k edges. Thus, k' vertices cannot cover more than kk' edges.

□

A Faster FPT Algorithm

Here are all the steps in the algorithm:

Algorithm Kernelization-FPT(G, k)

1. Let S' be the vertices of G of degree $> k$. If $|S'| > k$, return FALSE.
2. Let $G' = G - S'$ and let $k' = k - |S'|$.
3. If G' has more than kk' edges, return FALSE
4. Let G'' be the graph obtained after removing isolated vertices from G'
5. Return VertexCoverFPT$(G'', k' = k - |S'|)$

Correctness:

- From observation on high degree vertices, all vertices in G of degree $> k$ are in the vertex cover.

- By Observations, if the graph G' has more than kk' edges, than G cannot have vertex cover of size $\leq k$.

- By Result 1, VertexCoverFPT(G'', k') correctly returns the outcome of whether G'' has a vertex cover of size $\leq k'$.

Complexity Analysis

1. Let S' be the vertices of G of degree $> k$. If $|S'| > k$, return FALSE.
2. Let $G' = G - S'$ and let $k' = k - |S'|$.
3. If G' has more than kk' edges, return FALSE
4. Let G'' be the graph obtained after removing isolated vertices from G'
5. Return VertexCoverFPT$(G'', k' = k - |S'|)$

- Step 1 takes $O(|V| + |E|)$ time

- Step 2 takes $O(|V| + |E|)$ time

- Step 3 takes $O(|V| + |E|)$ time

- Step 4 takes $O(|V| + |E|)$ time

Consider the graph G'' obtained in Step 4.

G'' has at most $kk' \leq k^2$ edges.

Since G'' has no isolated vertices, it has $\leq 2k^2$ vertices.

Graph G'' is the 'small' kernel for the vertex cover problem. We can execute an exponential time algorithm on G''.

By Result 1, in Step 5, execution of VertexCoverFPT(G'', k') takes $O(k^2 \times 2^k)$ time.

Result 2

Let $G = (V, E)$ be a simple undirected graph that has a vertex cover of size at most k. Vertex cover problem admits a kernel consisting of $O(k^2)$ vertices and $O(k^2)$ edges. We can find the minimum vertex cover of G in $O(|V| + |E| + k^2 2^k)$ time.

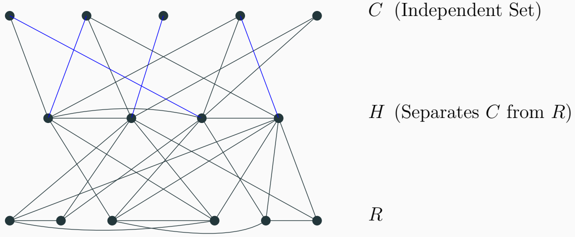
Crown Decomposition

Crown Decomposition - A general kernelization technique

Crown Decomposition of G

Crown decomposition of a graph $G = (V, E)$ is a partitioning of the set of vertices V in three disjoint sets $V = C \cup H \cup R$ such that

1. There is no edge between vertices in C and R. H separates C from R.
2. C is a non-empty independent set.
3. There is a matching of size $|H|$ in the bipartite graph induced between the vertices in C and H. I.e. the matching saturates the vertices in H.



Main Lemma

Let $G = (V, E)$ be a graph with at least $3k + 1$ vertices, none of them are isolated. In polynomial time we can determine either G has a matching of size at least $k + 1$ or find its crown decomposition.

Proof: We can use any of the matching algorithms to determine whether G has a matching of size $\geq k + 1$ in polynomial time. Assume that all possible matchings have fewer than $k + 1$ edges.

1. Let M be a maximal matching of G. Let V_M be the set of vertices corresponding to edges in M. The vertices $I = V \setminus V_M$ forms an independent set.
2. Consider the bipratite graph $B(V_M, I)$ consisting only of edges between V_M and I in G.
3. Let M' be a maximum matching in B and let X be a minimum vertex cover of B.
4. $|X| = |M'| \leq k$, as B is bipartite graph and maximum matching in G has $< k + 1$ edges (by assumption).

Claim

5. Claim

5. $X \cap V_M \neq \emptyset$.

Proof: Suppose not. I.e. $X \cap V_M = \emptyset$.

$\implies X \subseteq I$

We claim that $X = I$. If so, $|V_M| + |I| \leq 2k + k = 3k$, and that contradicts the fact that G has at least $3k + 1$ vertices and thus it can't be that $V_M \cap X = \emptyset$.

To complete this part of the argument, suppose $X \neq I$.

Let $v \in I \setminus X$.

Since no vertex of G is isolated, there is an edge uv incident on v where $u \in V_M$.

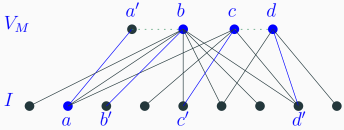
But to cover the edge uv, we need to have $u \in X$.

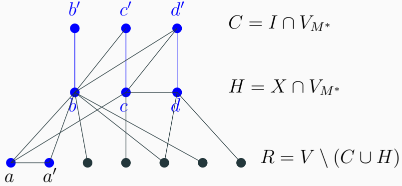
But we assumed that $V_M \cap X = \emptyset$. $\square$

Proof of Lemma (contd.)

Now we have that $X \cap V_M \neq \emptyset$.

6. Since $|X| = |M'|$, exactly one end of each edge of M' is in X. Let $M^* \subseteq M'$ such that every edge in M^* has one end point in $X \cap V_M$. Let V_{M^*} be the union of all vertices defining the edges in M^*.





$X = \{a, b, c, d\}$

$M' = \{aa', bb', cc', dd'\}$

$M^* = \{bb', cc', dd'\}$ $\qquad V_{M^*} = \{b, b', c, c', d, d'\}$

7. Define the sets C, H, and R for the crown decomposition as follows: $H = X \cap V_{M^*}$; $C = I \cap V_{M^*}$; $R = V \setminus (H \cup C)$

Observations on C, H, and R

Crown Set C

The set $C = I \cap V_{M^*}$ is a non-empty independent set.

Proof: C is independent as I is independent. $C \neq \emptyset$ as $X \cap V_M \neq \emptyset$, and each edge in the matching M' contributes exactly one end point to the vertex cover X of $B(V_M, I)$. □

Head Set H

The set $H = X \cap V_{M^*}$ separates C from R. Moreover, the induced bipartite graph on $C \cup H$ has a matching of size $|H|$.

Proof: For any vertex $v \in C = I \cap V_{M^*}$, $\exists u \in H = X \cap V_{M^*}$ such that $uv \in M^* \subseteq M'$ (and $u \in X$) $\implies v \notin X$ as for any edge $uv \in M'$ exactly one of its ends is in X.
Thus, $C \cup H$ has a matching of size $|H|$.
Since $v \in I$ and $v \notin X$, all neighbors of v in $B(V_M, I)$ are in $X \cap V_{M^*} = H$. □

Main Lemma

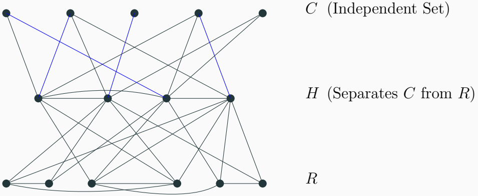
Let $G = (V, E)$ be a graph with at least $3k + 1$ vertices, none of them are isolated. In polynomial time we can determine either G has a matching of size at least $k + 1$ or find its crown decomposition.

Observe that the main computational steps are:

- Finding a maximum matching in G

- Finding the sets C, H and R.

Each step can be implemented in polynomial time.



Small Kernel for Vertex Cover using Crown Decomposition

Let $G = (V, E)$ be the given graph and let k be an integer parameter.
Question: Is there a vertex cover of size at most k?

We use the crown decomposition to find a small kernel as follows:

Algorithm VC-Kernel$\langle G, k \rangle$

1. Remove isolated vertices from G.
2. If G has $\leq 3k$ vertices, output G as the kernel and terminate.
3. Apply Crown Lemma on G. Either it reports that matching in G has $\geq k+1$ edges ($\implies$ G has a vertex cover of size $> k$) or a partitioning $V = C \cup H \cup R$.
4. Place all vertices in H in the vertex cover, and execute VC-Kernel$\langle G - H, k - |H| \rangle$.

Correctness of Algorithm VC-Kernel

The algorithm reports whether $G = (V, E)$ has a vertex cover of size $> k$ or outputs a kernel of size $\leq 3k$.

Proof: If $\exists$ matching of size $\geq k + 1$, the vertex cover of G requires $\geq k + 1$ vertices. Otherwise, consider the crown decomposition $V = C \cup H \cup R$.

- Recall, C is independent. $H \neq \emptyset$. H separates C from R. $\exists$ matching of size $|H|$ in the bipartite graph formed by C and $H \implies H$ is a vertex cover of induced graph of $C \cup H$.

- Graph $G - H$ consists of isolated vertices in C, and possibly some isolated vertices in the set R. They will be removed in the next call to VC-Kernel$\langle G - H, k - |H|\rangle$.

$\implies$ Crown decomposition reduces the problem to finding a vertex cover of size $\leq k - |H|$ in graph $G - H$. As $H \neq \emptyset$, $G - H$ is a smaller graph.

- Recursion terminates when G has fewer than $3k + 1$ vertices. $\square$

Kernel from Linear Program

Integer Linear Program for Vertex Cover

Integer LP for Vertex Cover

Let $G = (V, E)$ be the given graph. Associate an indicator $0 - 1$ variable x_v for each vertex $v \in V$ that indicates whether v is in the cover or not. The LP is given by

$$\text{Objective Function:} \qquad \text{minimize} \sum_{v \in V} x_v$$

$$\text{Subject to:} \quad \forall e = (uv) \in E : x_u + x_v \geq 1$$

$$x_v \in \{0, 1\}$$

Observation

Above ILP results in a vertex cover. Each edge is covered because of the constraint $x_u + x_v \geq 1$, and at least one of u or v has to be 1 indicating that the corresponding vertex is in the cover.

Since ILP's are NP-Hard, we relax it and solve the relaxed LP in polynomial time.

Relaxed LP for Vertex Cover

Objective Function: minimize $\sum_{v \in V} x_v$

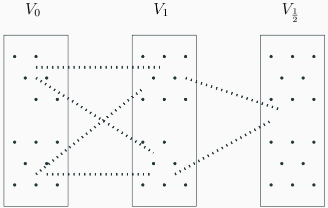
Subject to: $\forall e = (uv) \in E : x_u + x_v \geq 1$

$$\mathbf{0 \leq x_v \leq 1}$$

Note: Variables x_v's can take fractional values. The value of the objective function of the relaxed LP is a lower bound on the size of the vertex cover.

Three Sets

Define three sets of vertices based on LP values of variables x_v's:
$V_0 = \{v \in V | x_v < \frac{1}{2}\}$, $V_1 = \{v \in V | x_v > \frac{1}{2}\}$, and $V_{\frac{1}{2}} = \{v \in V | x_v = \frac{1}{2}\}$.



Observations

1. V_0, V_1, and $V_{\frac{1}{2}}$ is a partition of V, i.e. $V = V_0 \cup V_1 \cup V_{\frac{1}{2}}$
2. The set V_0 is an independent set.
3. There are no edges between vertices in V_0 and $V_{\frac{1}{2}}$.

Nemhauser-Trotter theorem

$V_0 = \{v \in V | x_v < \frac{1}{2}\}$, $V_1 = \{v \in V | x_v > \frac{1}{2}\}$, and $V_{\frac{1}{2}} = \{v \in V | x_v = \frac{1}{2}\}$.

Theorem

There is a minimum vertex cover $S \subseteq V$ of G such that $V_1 \subseteq S \subseteq V_1 \cup V_{\frac{1}{2}}$

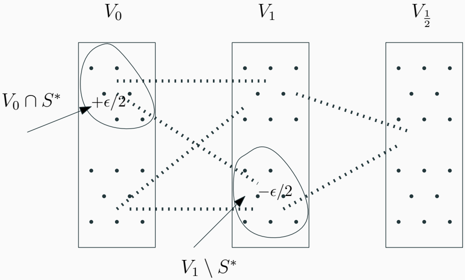
Proof: Let S^* is a minimum vertex cover of G.
- Define $S = (S^* \setminus V_0) \cup V_1$.
- S is a vertex cover of G as any vertex in V_0 is only adjacent to vertices in V_1.

Using contradiction, we show that S forms a minimum vertex cover.

- Assume $|S| > |S^*|$
- Observe that $|S| = |S^*| - |S^* \cap V_0| + |V_1 \setminus S^*|$
$\implies |V_1 \setminus S^*| > |S^* \cap V_0|$ as we assumed $|S| > |S^*|$.
- Now we will construct another feasible solution of the relaxed LP that has a smaller optimum value contradicting the optimality of LP.

Nemhauser-Trotter theorem (contd.)

Define $\epsilon = \min\{|x_v - \frac{1}{2}|, v \in V_0 \cup V_1\}$.



Modify x_v values as follows:

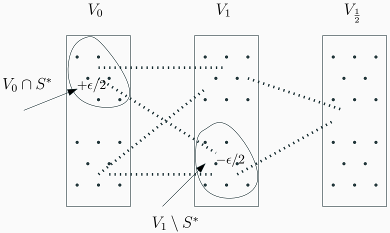
- For all vertices $v \in V_1 \setminus S^*$: set $y_v = x_v - \frac{\epsilon}{2}$.
- For all vertices $v \in V_0 \cap S^*$: set $y_v = x_v + \frac{\epsilon}{2}$.
- For all remaining vertices: set $y_v = x_v$
- Note that $\sum x_v > \sum y_v$, as we had $|V_1 \setminus S^*| > |S^* \cap V_0|$.

- Next we show that y_v values satisfy the constraints of relaxed LP

$\implies$ x_v's are not optimal and that leads to a contradiction to optimality of LP.

Nemhauser-Trotter theorem (contd.)

- Consider any edge $e = (uv) \in G$. We need to show that $y_u + y_v \geq 1$.
- Consider the cases where one of the end vertices of any edge is in $V_0 \cap S^*$ or $V_1 \setminus S^*$, as for all other edges $y_u + y_v = x_u + x_v \geq 1$.



(A) Suppose $u \in V_0 \cap S^*$: v can only be in V_1. If $v \in V_1 \setminus S^*$, $x_u + x_v = y_u + \epsilon/2 + y_v - \epsilon/2 = y_u + y_v \geq 1$.
If $v \in V_1 \cap S^*$, $y_u + y_v = x_u + \epsilon/2 + x_v \geq x_u + x_v \geq 1$.

(B) $u \in V_1 \setminus S^*$: If $v \in V_0$, a similar argument applies. If $v \in V_{\frac{1}{2}}$, $y_u + y_v = x_u - \epsilon/2 + x_v \geq 1$ as $x_v = \frac{1}{2}$ and $x_u > \frac{1}{2} + \epsilon/2$. $\square$

Applying Nemhauser-Trotter theorem to vertex cover

We know that an optimal vertex cover S satisfies $V_1 \subseteq S \subseteq V_{\frac{1}{2}} \cup V_1$.
Perform the following steps to determine if G has a vertex cover of size $\leq k$.

Step 1: If value returned by relaxed LP is $> k$. Report G has vertex cover of size $> k$ and Stop.

Step 2: Include V_1 in the vertex cover and determine if $G \setminus (V_0 \cup V_1)$ has a vertex cover of size $\leq k - |V_1|$.

Reduced graph $G \setminus (V_0 \cup V_1)$

G has a vertex of size $\leq k$ if and only if $G \setminus (V_0 \cup V_1)$ has a vertex cover of size $\leq k - |V_1|$.

Proof: We know that there is a minimum vertex cover S of G such that $V_1 \subseteq S \subseteq V_{\frac{1}{2}} \cup V_1$. If S is a vertex cover of size $\leq k$ for G, $\implies S \setminus V_1$ is a vertex cover of size $\leq k - |V_1|$ for the graph induced by $V \setminus (V_0 \cup V_1) = V_{\frac{1}{2}}$.

For the other direction, observe that the graph induced by V_0 is isolated, and only has edges to the vertices in the set V_1. If S' is a vertex cover of the graph induced by $V_{\frac{1}{2}}$, $S' \cup V_1$ is a vertex cover of G. $\square$

Cardinality of $V_{\frac{1}{2}}$

$|V_{\frac{1}{2}}| \leq 2k$.

Proof: By definition, the linear program has assigned each variable $x_v \in V_{\frac{1}{2}}$ the value of $\frac{1}{2}$. Thus,

$$\begin{aligned} |V_1| &= \sum_{v \in V_{\frac{1}{2}}} 2x_v \\ &\leq 2 \sum_{v \in V} x_v \\ &\leq 2k \qquad \square \end{aligned}$$

Small Kernel for Vertex Cover

Lemma

The induced graph on the vertices in $V_{\frac{1}{2}}$ forms a kernel for the vertex cover problem consisting of at most $2k$ vertices. Moreover, we can determine the kernel in polynomial time.

Proof:

- Linear programs can be solved in polynomial time and we can determine if its objective value $\leq \frac{1}{2}$.

- We can form the sets V_0, V_1, and $V_{\frac{1}{2}}$ in $O(|V|)$ time.

- Computation of the induced graph on $V_{\frac{1}{2}}$ takes $O(|V| + |E|)$ time.

- Thus we can determine the kernel of size $\leq 2k$ of G in polynomial time provided that it has a vertex cover of size $\leq k$.

□

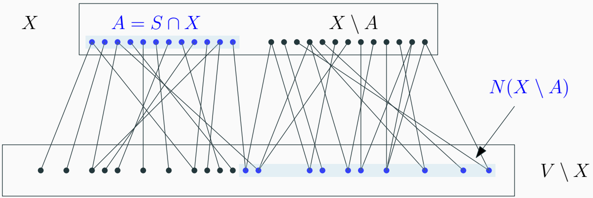
Iterative Compression

Illustration via Vertex Cover

A Property of Vertex Cover

Let $X, S \subseteq V$ be two vertex covers of $G = (V, E)$. Let $A = S \cap X$, and let $N(X \setminus A)$ represent neighbors of vertices in $X \setminus A$ in the set $V \setminus X$. The set $Y = A \cup N(X \setminus A)$ is a vertex cover of G if the graph induced by vertices in $X \setminus A$ is an independent set.

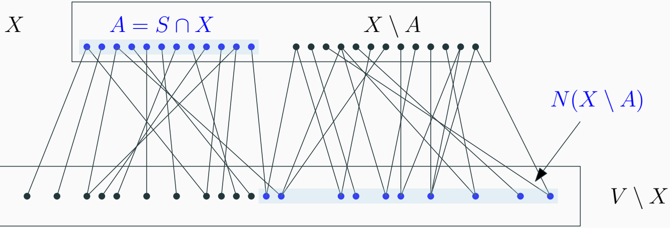
Proof: X is a vertex cover $\implies V \setminus X$ is an independent set.
$A \subseteq Y \implies$ all edges incident to A are covered.
$N(X \setminus A) \subseteq Y \implies$ all edges incident to $N(X \setminus A)$ are covered.
If $X \setminus A$ is independent, Y is a vertex cover of G. □



Large VC → Small VC

Input: $X \subseteq V$, $G = (V, E)$, $|X| = k + 1$, and X is a vertex cover of G.
Output: Does G contain a vertex cover of size $\leq k$?

Idea: Select an arbitrary subset $A \subset X$ of $\leq k$ vertices. Check whether there exists a vertex cover $S \supseteq A$ consisting of k vertices.



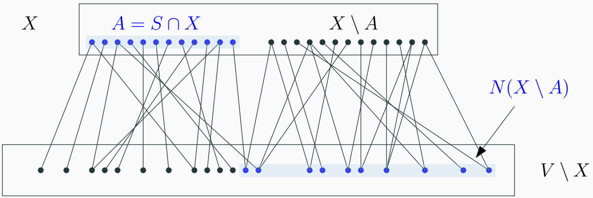
Large VC → Small VC

Observation

Let $N(X \setminus A)$ represent the neighbors of $X \setminus A$ in $V \setminus X$. Set $S = A \cup N(X \setminus A)$. S is the required vertex cover of G if

1. $|S| \leq k$.
2. There are no edges in the graph induced by $X \setminus A$.

Given X, we can try all possible subsets A of X.
$\#$ subsets A of $X = O(2^k)$, and for each subset we can test the required conditions in $O(|V| + |E|)$ time.



Compression algorithm for testing whether G has a vertex cover of size $\leq k$:

Step 1: Consider an arbitrary permutation of vertices of G. Let it be $v_1, \ldots, v_n$.

Step 2: Let G_k be the graph induced by vertices $V_k = \{v_1, \ldots, v_k\}$. Note that $X = V_k$ is a vertex cover of G_k of size k.

Step 3: For $i := k+1$ to n do

1. Compute G_i by adding the vertex v_i and all of its incident edges to G_{i-1}. Note that $V_i = \{v_1, \ldots, v_i\}$.
2. Set $X \leftarrow \{v_i\} \cup X$. Note that X is a vertex cover of G_i.
3. If $|X| = k+1$, check whether there exists a vertex cover $S \subset V_i$ of size $\leq k$ for G_i. If so, set $X \leftarrow S$, otherwise report G doesn't have a vertex cover of size $\leq k$.

Claim

The above procedure takes $O(2^k|V|(|V|+|E|))$ time to determine whether G has a vertex cover of size $\leq k$.

Proof: Note that $G = G_n$.

- At the start of the iteration $i \in \{k+1, n\}$, we know that X is a vertex cover of size $\leq k$ for the graph G_{i-1}.

- If $|X \cup v_i| \leq k$, we already have a vertex cover of size $\leq k$ for G_i.

- Otherwise, we apply the observation as X is a vertex cover of G_i consisting of $k+1$ vertices and we are seeking a vertex cover S of size at most k. We consider all possible subsets A of size $\leq k$ of X and determine whether there exists $S \supset A$ consisting of $\leq k$ vertices that covers G_i.

- Outcome is either we find a set S, or we fail. If we find S, we set $X \leftarrow S$ and proceed to the next iteration. If we fail, G can't have a vertex cover of size $\leq k$ as its subgraph G_i doesn't admit vertex cover of size $\leq k$.

- Running time for each iteration is $O(2^k(|V|+|E|)$. $\square$

Iterative Compression via FVS

Feedback Vertex Set (FVS)

Let $G = (V, E)$ be a simple undirected graph. A subset $S \subset V$ of vertices is called a feedback vertex set if the graph induced on the vertices $V \setminus S$ (denoted by $G(V \setminus S)$) is acyclic.

FVS Decision Problem: Does G contain a FVS of size at most k?

We will first look into a specific version of FVS problem, and then show how an iterative compression technique can be applied to answer the decision problem.

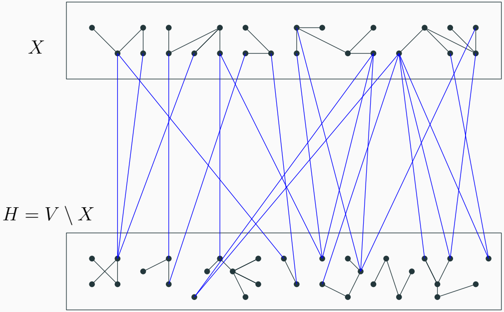
Disjoint Feedback Vertex Set Problem

Input consists of $G = (V, E)$, a parameter k, a FVS $X \subset V$ of size $k + 1$. Decide whether G has FVS $S \subseteq V \setminus X$ of size $\leq k$? We denote this problem as D-FVS(G, X, k).

Disjoint Feedback Vertex Set Problem

$G(V \setminus X)$ and $G(X)$ are forests

- If X is FVS of $G = (V, E)$, the graph, $G(H = V \setminus X)$, induced on the vertices $H = V \setminus X$ is a forest.
- Some $S \subset H$ can be FVS of G provided that the graph, $G(X)$, induced on the vertices of X is acyclic.



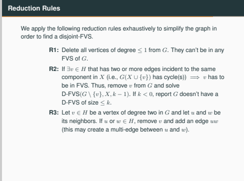
Reduction Rules

We apply the following reduction rules exhaustively to simplify the graph in order to find a disjoint-FVS.

R1: Delete all vertices of degree ≤ 1 from G. They can't be in any FVS of G.

R2: If $\exists v \in H$ that has two or more edges incident to the same component in X (i.e., $G(X \cup \{v\})$ has cycle(s)) $\implies$ v has to be in FVS. Thus, remove v from G and solve D-FVS$(G \setminus \{v\}, X, k-1)$. If $k < 0$, report G doesn't have a D-FVS of size $\leq k$.

R3: Let $v \in H$ be a vertex of degree two in G and let u and w be its neighbors. If u or $w \in H$, remove v and add an edge uw (this may create a multi-edge between u and w).



1. In R3, if both $u, w \in X$, no shortcut is added.The reason is that none of vertices in X can be in D-FVS.
2. After rules R1-R3 are applied exhaustively, each vertex in H has degree at least 2. For all non-isolated vertices in $G(H)$, their degree is ≥ 3.

Observation

Structure of the resulting graph

Let G be the graph obtained after applying the reduction rules R1-R3 exhaustively.

1. The number of connected components in $G(X)$ is $\leq k+1$.
2. Consider the forest $G(H)$ induced by vertices in H. For any isolated vertex in $G(H)$ its degree is ≥ 2 in G. For any non isolated vertex in $G(H)$, its degree is ≥ 3 in G.

Branching on degree ≤ 1 vertices of forest H

Perform the following branching steps for any degree ≤ 1 vertex v of $G(H)$

$v \in$ **D-FVS:** Execute D-FVS$(G \setminus \{v\}, X, k-1)$.

$v \notin$ **D-FVS:** Move v to the set X, merge the components in X that are adjacent to v, and execute D-FVS$(G, X \cup v, k)$.

Note: During each branching, we also apply the reduction rules.

Correctness of Branching

We make some observations about the branching process.

1. In each call to branching, we either reduce k by 1, or reduce the number of connected components in X by at least 1. Therefore, the branching process terminates in at most $2k+1$ steps, as there are $\leq k+1$ components in $G(X)$.
2. Moving a degree ≤ 1 vertex $v \in G(H)$ to X is safe as $G(X \cup \{v\})$ is acyclic. Otherwise, we would have applied the reduction rule R2.
3. If ever $k < 0$, we terminate and report that D-FVS(G, X, k) has no solution.
4. It is possible that we may reach a situation during branching where we have a single component in $G(X)$. Remember that we are still applying the reduction rules R1-R3, and that ensures what vertices will be added to D-FVS.

D-FVS Summary

D-FVS

Discrete feedback vertex set problem D-FVS(G, X, k) can be solved in $O(4^k n^{O(1)})$ time, where n is the number of vertices in G.

Proof:
- Rules R1-R3 can be implemented in polynomial time with respect to the size of G.
- Branching terminates in at most $2k+1$ steps, where in each step either we include a vertex v of degree ≤ 1 of $G(H)$ in D-FVS or exclude it.
- Thus, the branching tree has $2^{2k+1} = O(4^k)$ nodes.

□

Iterative Compression for FVS

Compression algorithm for testing whether G has FVS of size $\leq k$:

Step 1: Consider an arbitrary permutation of vertices of G. Let it be $v_1, \ldots, v_n$.

Step 2: Let G_k be the graph induced by vertices $V_k = \{v_1, \ldots, v_k\}$. Note that $X = V_k$ is a FVS of G_k of size k.

Step 3: For $i := k + 1$ to n do

1. Compute G_i by adding vertex v_i and all of its incident edges to G_{i-1}. Note: $V_i = \{v_1, \ldots, v_i\}$.
2. Set $X \leftarrow \{v_i\} \cup X$. Note that X is a FVS of G_i.
3. If $|X| = k + 1$, check whether there exists a FVS $S \subset V_i$ of size $\leq k$ for G_i. If so, set $X \leftarrow S$, otherwise report G doesn't have a FVS of size $\leq k$.
   To find S, we try all subsets $A \subset X$ and solve for D-FVS$(G(V_i) \setminus A, X \setminus A, k - |A|)$.

FVS Summary

D-FVS

For a given graph G and a parameter k, we can check in $5^k n^{O(1)}$ time whether G has a feedback vertex set of size at most k, where n is the number of vertices in G.

Proof: Recall that in the D-FVS(G, X, k) problem, the input consists of a graph G, a parameter k, a FVS $X \subset V$ of size $k+1$, and the problem is to decide whether G has FVS $S \subseteq V \setminus X$ of size $\leq k$?

Consider any iteration $i \geq k+1$ of the algorithm:
- We have the FVS $X \cup \{v_i\}$ of size $\leq k+1$ for $G_i \implies G_i(V_i \setminus X)$ is a forest.
- Our task is to decide if $\exists S \subset V_i$ of size $\leq k$ such that S is FVS of G_i.
- We make guess of which vertices of S are from X. Assume $A = S \cap X$.
- Consider the graph $G_i(V_i \setminus A)$.
- We want a FVS of size $\leq k - |A|$ in $G_i(V_i \setminus A)$ where all of its vertices are from the set $V_i \setminus X$.
This is precisely the D-FVS$(G(V_i) \setminus A, X \setminus A, k - |A|)$ problem.

FVS Analysis (contd.)

Next we analyze the running time.

In iteration $i \geq k+1$, we try all possible subsets A of X, where $|X| = k+1$, and for each subset A, we make a call to an appropriate D-FVS$(G(V_i) \setminus A, X \setminus A, k - |A|)$ problem.

We know that the running time for the D-FVS problem is $4^{k-|A|}n^{O(1)}$.

Therefore, the total running time for the i-th iteration is

$$\sum_{j=0}^{k} \binom{k+1}{j} 4^{k-j} n^{O(1)} = 5^k n^{O(1)}$$

Note that $(1+4)^k = \sum_{j=0}^{k} \binom{k+1}{j} 1^j \cdot 4^{k-j} = 5^k$.

Since i ranges from $k+1$ to n, the total running time for the FVS decision problem is $5^k n^{O(1)}$.

$\square$

References

1. Downey and Fellows, Parameterized Complexity. Springer, 1999.
2. Cygan, Fomin, Kowalik, Lokshtanov, Marx, Pilipczuk, Pilipczuk, and Saurabh, Parameterized Algorithms, Springer 2015.