KAIST, School of Computing, Spring 2024 | **Scribed By: Eunjung KIM**
Algorithms for NP-hard problems (CS492) | **Lecture #6**
Lecture: Eunjung KIM | **14 March 2024**

Contents

1 Iterative compression technique

1.1 $\mathcal{O}^*(5^k)$-time algorithm for FEEDBACK VERTEX SET

The *iterative compression* technique solves a parameterized problem P by *iteratively* solving a *compression* version of P. This is where the name comes from. We study this technique with an exemplary algorithm for FEEDBACK VERTEX SET. Consider the following compression version of FEEDBACK VERTEX SET.

COMPRESSION FVS
Instance: a graph $G = (V, E)$, a feedback vertex set $X \subseteq V$.
Parameter: $|X|$
Question: Does G have a feedback vertex set Y such that $|Y| < |X|$?

Suppose that you're given a fvs X_i of size at most k for a graph G_i. If G_i expands to a slightly bigger graph G_{i+1}, where G_{i+1} contains precisely one more vertex v_{i+1} on top of G_i, then we know that $X_i \cup \{v_{i+1}\}$ is again a fvs of G_{i+1}. But its size might exceed our allowed budget k (if not, $X_i \cup \{v_{i+1}\}$ is a trivial solution to COMPRESSION FVS). Our goal is to look for an alternative fvs of G_{i+1} of size at most k, that is, a fvs whose size is strictly smaller than the given fvs. This extra information of a fvs $X_i \cup \{v_{i+1}\}$ of small size at hand, albeit a bit exceeding our budget, makes the task of algorithm design way easier.

Lemma 1. *If there is an algorithm $\mathcal{A}$ of* COMPRESSION FVS *running in time $c^{|X|} \cdot n^d$, then there is an algorithm for* FEEDBACK VERTEX SET *running in time $c^k \cdot n^{d+1}$.*

Proof: Let $v_1, \ldots, v_n$ be the vertices of G. For each $1 \leq i \leq n$, we define $G_i = G[\{v_1, \ldots, v_i\}]$, that is, G_i is the subgraph of G induced by the first i vertices. Suppose that X_i is a fvs of G_i of size at most k. Such X_i exists for i up to k. For $i + 1 \leq n$, note that $X_i \cup \{v_{i+1}\}$ is a fvs of G_{i+1} and $(G_{i+1}, X_i \cup \{v_{i+1}\})$ is a legitimate instance to COMPRESSION FVS. Now we run the algorithm $\mathcal{A}$ on $(G_{i+1}, X_i \cup \{v_{i+1}\})$. If $\mathcal{A}$ returns a fvs X_{i+1} of G_{i+1} of size at most k, we can proceed to the next iteration for $i + 2$ or declare it as a fvs of G in case $i + 1 = n$. On the other hand, if $\mathcal{A}$ returns NO, then this means that not only G_{i+1} is a NO-instance but G_n is a NO-instance as well: indeed, if G_n has a feedback vertex set X_n of size at most k, then $X_n \cap \{v_1, \ldots, v_{i+1}\}$ is a feedback vertex set of G_{i+1} and its size is clearly at most k. Therefore, we can correctly return NO as an output of the algorithm. To see the running time, notice that the aforementioned algorithm executes $\mathcal{A}$ at most n times. $\square$

Thanks to Lemma 1, now we can focus on designing an efficient fpt-algorithm for COMPRESSION FVS[1]. We expect that designing an algorithm for COMPRESSION FVS would be easier than designing an algorithm for FEEDBACK VERTEX SET because the latter problem is at least as hard as the former. In fact, COMPRESSION FVS can be even further reduced to the following variant of FEEDBACK VERTEX SET.

DISJOINT FVS
Instance: a graph $G = (V, E)$, a feedback vertex set $\tilde{X} \subseteq V$, and an integer k.
Question: Does G have a feedback vertex set $\tilde{Y}$ such that $|\tilde{Y}| < |\tilde{X}|$ and $\tilde{Y} \cap \tilde{X} = \emptyset$?
Parameter: k.

[1]Instead of using iterative compression, we can obtain an approximate feedback vertex set of size at most $2k$ using a 2-approximation algorithm for FEEDBACK VERTEX SET and apply the algorithm for COMPRESSION FVS at most k times. That is, starting from X, we obtain a smaller solution if possible and feed it to the next instance of COMPRESSION FVS. The running time will be $\mathcal{O}^*(c^{|X|} \cdot n^d \cdot k)$ in this case.

The basis of reducing[2] Compression FVS to Disjoint FVS is to rewrite a feasible solution Y as a disjoint union of two sets $I := Y \cap X$ and $\tilde{Y} := Y \setminus X$. Furthermore, if $|Y| < |X|$ then $|Y \setminus X| < |X \setminus Y|$. So, in order to find a solution Y to Compression FVS, we can 'guess' $Y \cap X$ by enumerating all subsets I of X, remove the guessed part I from G, and then find a fvs $\tilde{Y}$ of $G - I$ such that $\tilde{Y}$ is disjoint from $X \setminus I$ and has strictly smaller size than $X \setminus I$.

Algorithm 1 Algorithm for Disjoint FVS

```
 1: procedure dfvs($G, \tilde{X}, k$)
 2:     Let $F = G[V \setminus \tilde{X}]$ and $\mathcal{C}$ be the set of connected components of $G[\tilde{X}]$.
 3:     Delete all vertices having degree 1 in $G$.
 4:     If $v \in F$ has degree 2 in $G$, bypass it (delete $v$ and connect its neighbors, possibly identical).
 5:     if $G$ is acyclic then return $\emptyset$.
 6:     else if $G[\tilde{X}]$ contains a cycle then return No.
 7:     else if $G$ contains a cycle and $k = 0$ then return No.
 8:     Choose a leaf $v$ of $F$.                                   $\triangleright\ k > 0$
 9:     if $v$ has two neighbors in a single components of $\mathcal{C}$ then
10:         return dfvs($G - v, \tilde{X}, k - 1) \cup \{v\}$
11:     else                  $\triangleright\ v$ has two neighbors belonging to distinct components of $\mathcal{C}$
12:         return dfvs($G - v, \tilde{X}, k - 1) \cup \{v\}$ or dfvs($G, \tilde{X} \cup \{v\}, k$).
```

Lemma 2. *The algorithm* **dfvs**, *given an instance* $(G, \tilde{X}, k)$, *solves* Disjoint FVS *correctly in time* $\mathcal{O}^*(2^{\mu(I)})$, *where* $\mu(I) = k + |cc(G[\tilde{X}])|$.

Proof: We omit the correctness proof (which is rather straightforward, see the textbook by Cygan et. al. for details). To analyze the running time, we introduce a measure $\mu(I)$ of an instance $I = (G, \tilde{X}, k)$ to Disjoint FVS.

$$\mu(G, \tilde{X}, k) = k + |cc(G[\tilde{X}])|.$$

In Line 12, each branching decreases the measure μ by at least one. Indeed, $\mu(G - v, \tilde{X}, k - 1) = k - 1 + |cc(G[\tilde{X}])| = \mu(I) - 1$, and the measure decreases by one in the first branching. In the second branching, recall that v is adjacent with (at least) two distinct components of $G[\tilde{X}]$ and thus by adding v to $\tilde{X}$, we decreases the number of connected components by at least one. That is, $\mu(G, \tilde{X} \cup \{v\}, k) \leq \mu(I) - 1$. From $\mu(I) \geq 1$, the depth (as the number of branching nodes where Line 12 is invoked) of a search tree algorithm is at most $\mu(I)$ and the running time follows. □

Lemma 3. *There is an algorithm $\mathcal{B}$ for* Compression FVS *running in time* $5^{|X|} \cdot n^d$.

Proof: Given an instance (G, X) to Compression FVS, we create an instance $(G', \tilde{X}, k')$ to Disjoint FVS for every $I \subsetneq X$ as follows:

$$G' = G - I, \tilde{X} = X \setminus I \text{ and } k' = |\tilde{X}| - 1.$$

The algorithm $\mathcal{B}$ on the input instance (G, X) is described below.

We first observe that if an instance $(G', \tilde{X}, |\tilde{X}| - 1)$ of Disjoint FVS is a Yes-instance at Line 3, then the output $\tilde{Y} \cup I$ is indeed a solution to (G, X) for Compression FVS. Indeed, $|\tilde{Y}| \leq |\tilde{X}| - 1$ implies that

[2]Creating a connection between the two problems so that by solving instances of the latter, one can obtain a solution to the former.

Algorithm 2 Algorithm for COMPRESSION FVS

```
1: procedure $\mathcal{B}(G, X)$
2:     for all $I \subsetneq X$ do
3:         if $\mathbf{dfvs}(G', \tilde{X}, |\tilde{X}| - 1) \neq$ NO then
4:             Let $\tilde{Y} = \mathbf{dfvs}(G', \tilde{X}, |\tilde{X}| - 1)$ and return $\tilde{Y} \cup I$
5:     return NO
```

$|\tilde{Y}| + |I| < |\tilde{X}| + |I| = |X|$. Moreover, $\tilde{Y} \cup I$ is a fvs of G because of $G - (I \cup \tilde{Y}) = (G - I) - \tilde{Y} = G' - \tilde{Y}$; $G' - \tilde{Y}$ is acyclic as $\tilde{Y}$ is a fvs of G'.

Therefore, to see the correctness of the algorithm $\mathcal{B}$ we only need to settle the claim:

if $\mathcal{B}$ returns NO, then (G, X) is a NO-instance to COMPRESSION FVS.

If (G, X) is a YES-instance to COMPRESSION FVS, let Y be a fvs of G such that $|Y| < |X|$. Then for $I := Y \cap X$, the corresponding instance $(G', \tilde{X}, k')$ defined as $G' : G - I$, $\tilde{X} := X \setminus I$ and $k' := |\tilde{X}| - 1$, the vertex set $\tilde{Y} := Y \setminus I$ is a fvs of G': indeed $G' - \tilde{Y} = G - I - \tilde{Y} = G - Y$ is acyclic due to the assumption that Y is a fvs of G. Moreover, $|\tilde{Y}| + |I| = |Y| < |X| = |\tilde{X}| + |I|$ implies that $|\tilde{Y}| < |\tilde{X}|$. Clearly, $\tilde{Y}$ is disjoint from $\tilde{X}$. Therefore, $\tilde{Y}$ is a solution to $(G', \tilde{X}, |\tilde{X}| - 1)$ for DISJOINT FVS. In particular, there exists some I^*(not necessarily the same I) such that the corresponding instance of DISJOINT FVS is YES, and therefore the condition of Line 3 is satisfied. Accordingly, the output of $\mathcal{B}(G, X)$ is not NO. This proves the correctness of the algorithm $\mathcal{B}$.

Finally, we observe that for all $I \subsetneq X$ of size i, an instance I to DISJOINT FVS with $\mu(I) = |X| - i - 1 + |cc(G[\tilde{X}])| \leq 2(|X| - i) - 1$ is created. For each such I, the algorithm $\mathbf{dfvs}$ runs in time $\mathcal{O}^*(2^{\mu(I)})$, thus in $\mathcal{O}^*(4^{|X|-i})$ time. Therefore, the algorithm $\mathcal{B}$ runs in time

$$\sum_{i=0}^{|X|-1} \binom{|X|}{i} \cdot 4^{|X|-i} \cdot n^d \leq (4+1)^{|X|} \cdot n^d.$$

□

2 Simple randomized algorithms

2.1 Feedback Vertex Set

We present a randomized algorithm for Feedback Vertex Set running in time $O^*(4^k)$. We first apply reduction rules first.

Reduction Rule 1: If u has has a loop in G, then delete v and decrease k by one.
Reduction Rule 2: If u has degree at most 1 in G (and does not have a loop), then delete v.
Reduction Rule 3: If u has degree 2 with neighbors v, w in G (possibly $v = w$), by delete u and add an edge (v, w).

Notice that application of Reduction Rule 3 may create parallel edges and loops - cycles of length two and one. Hence, G is a graph with parallel edges in the remainder of this subsection. The degree of a vertex is the number of incident edges, not the number of neighbors. Provided Reduction Rules 0-2 have been applied exhaustively, we can assume that G has minimum degree at least three.

The key observation behind the randomized $O^*(4^k)$-algorithm is sparsity of a forest. Let S be a feedback vertex set of G. Then $G - S$ have at most $|V(G) \setminus S| - 1$ edges, which accounts for at most two among the minimum degree 3 of the vertices in $V(G) \setminus S$. Hence, more than $|V(G)| - |S|$ edges are lying between S and $V(G) \setminus S$. This is formalized in the lemma below.

Lemma 4. *Let G be a graph with minimum degree three. Then for any feedback vertex set S, at least half of E are incident with S.*

Proof: We let $F := V(G) \setminus S$. Let us denote by $E(S)$ the set of edges whose both endpoints are in S and by $E(S, F)$ the set of edges which has precisely one endpoint in each of S and F. Let us count the sum $\sum_{v \in F} deg(v)$ in a different way. The only edges that contribute to this sum are $E(S, F) \cup E(F)$. Observe that an edge of $E(S, F)$ counts precisely once in this sum, and an edge of $E(F)$ is counted twice. Therefore, with the minimum degree condition on V it holds that

$$\sum_{v \in F} deg(v) = |E(S, F)| + 2|E(S)| \geq 3|F|$$

It follows that

$$|E(S, F)| \geq 3|F| - 2|E(F)| \geq 3(|E(F)| + 1) - 2|E(F)| > |E(F)|,$$

where the second inequality is due to the fact that the number of edges in a tree T is at most the number of vertices of T minus one. Observe that the edge set incident with S is $E(S) \cup E(F, S)$. From

$$|E(S)| + |E(S, F)| = \frac{1}{2}(2|E(S)| + 2|E(S, F)|) > \frac{1}{2}(|E(S)| + |E(S, F)| + |E(F)|) = \frac{1}{2}|E|,$$

we know that at least half of the edge set E is incident with S. This complete the proof. □

Thanks to Lemma 4, a randomly chosen edge e is incident with a (prescribed) feedback vertex set S with probability at least $\frac{1}{2}$. By again randomly choosing one endpoint of e, we choose one of S with probability at least $\frac{1}{4}$.

Algorithm 3 Algorithm for FEEDBACK VERTEX SET

```
1: procedure FVS($(G, k)$)
2:     Apply Reduction Rules 2-3 exhaustively.                ▷ $k$ remains the same
3:     if $G$ contains a cycle and $k = 0$ then return NO and terminate.
4:     else if $k \geq 0$ and $G$ is acyclic then return $\emptyset$.
5:     else if $G$ has a loop at some vertex $v$ then return FVS$(G - v, k - 1) \cup \{v\}$.
6:     else        ▷ $G$ is loopless, has a cycle, $k > 0$ and $deg(v) \geq 3$ for every $v \in V$.
7:         Pick an edge $e$ uniformly at random.
8:         Pick an endpoint of $e$ uniformly at random. Let $v$ be the chosen vertex.
9:         return FVS$(G - v, k - 1) \cup \{v\}$.
```

Lemma 5. *On an input instance (G, k) to* FEEDBACK VERTEX SET*, the procedure* FVS(G, k)

(i) runs in polynomial time,

(ii) outputs either No *or a feedback vertex set of G of size at most k,*

(iii) outputs a (feedback) vertex set of size at most k with probability at least $\frac{1}{4^k}$ if (G,k) is Yes.

Proof: The running time is straightforward. Before proceeding with the proof of (ii)-(iii), we point out that any input (G',k') to the procedure FVS for the subsequent calls incurred by $\mathsf{FVS}(G,k)$ is a legitimate instance of Feedback Vertex Set: that is, $k' \geq 0$. Indeed, we decrease the parameter k by one every time we make a call to FVS, and when $k=0$ an output is returned at Lines 3-4, which means no subsequent call is made.

We prove (ii) by induction on k. It suffices to prove that if $\mathsf{FVS}(G,k)$ returns a vertex set S, then S is a feedback vertex set of G of size at most k. When $k=0$, the fact that some vertex set S is returned means that (G,k) does not satisfy the condition of Line 3 and thus G is acyclic. Now that (G,k) meets the condition of Line 4, we know that $S=\emptyset$. It is clear that $S=\emptyset$ is a feedback vertex set of an acyclic graph G of size at most $k=0$. Consider $k>0$ and notice that S is returned at either Line 5 or 9. Especially, this means that the output of $\mathsf{FVS}(G-v,k-1)$ is $S\setminus v$. By induction hypothesis, $S\setminus v$ is a feedback vertex set of $G-v$ of size at most $k-1$. Hence, $G-v-S\setminus v$ is acyclic and thus S is a feedback vertex set of G. Clearly $|S|\leq k$. This proves (ii).

Now we prove (iii). Suppose that (G,k) is a Yes-instance. If G is acyclic, then (iii) trivially holds with probability 1. In particular, G is always acyclic when $k=0$ due to the assumption that (G,k) is a Yes-instance. Therefore, we may assume that G has a cycle and $k>0$. We claim that the input $(G-v,k-1)$ to a subsequent call at Line 5 or 9 is Yes with probability at least $\frac{1}{4}$. If G has a loop at v, then v must be included in any solution, and $(G-v,k-1)$ is again a Yes-instance with probability 1. If G does not have a loop, then Line 8 chooses a vertex v contained in a solution S of size at most k with probability $\frac{1}{4}$. Indeed, Lemma 4 implies that the edge e chosen at Line 7 is in $E(S)\cup E(S,V\setminus S)$ with probability 0.5. In case $e\in E(S)$, the probability that a random endpoint v of e is in S is 1. In case $e\in E(S,V\setminus S)$, the probability is 0.5. Therefore,

$$\begin{aligned}\mathsf{Pr}[v\in S] = \mathsf{Pr}[e\in E(S)\cup E(S,V\setminus S)]\times \mathsf{Pr}[v\in S|e\in E(S)\cup E(S,V\setminus S)]\\ \geq 0.5\times 0.5\end{aligned}$$

Notice that when $v\in S$, then $S\setminus v$ is a feedback vertex set of size at most $k-1$ of $G-v$. That is, $(G-v,k-1)$ is a Yes-instance. Therefore, the created instance $(G-v,k-1)$ at Line 9 is a Yes-intance with probability at least $\frac{1}{4}$, as claimed.

To finalize the proof of (iii), we recall that by induction hypothesis, $\mathsf{FVS}(G-v,k-1)$ returns a vertex set with probability at least $\frac{1}{4^{k-1}}$ when $(G-v,k-1)$ is Yes. Now[3],

$$\begin{aligned}&\mathsf{Pr}[\mathsf{FVS}(G,k)\text{ returns a vertex set}]\\ &\qquad\geq \mathsf{Pr}[\mathsf{FVS}(G,k)\text{ returns a vertex set at Line 9}]\\ &\qquad= \mathsf{Pr}[(G-v,k-1)\text{ is Yes and }\mathsf{FVS}(G-v,k-1)\text{ returns a vertex set}]\\ &\qquad\geq \mathsf{Pr}[(G-v,k-1)\text{ is Yes}]\\ &\qquad\quad\times \mathsf{Pr}[\mathsf{FVS}(G-v,k-1)\text{ returns a vertex set }|(G-v,k-1)\text{ is Yes}]\\ &\qquad\geq \frac{1}{4}\times\frac{1}{4^{k-1}}=\frac{1}{4^k}.\end{aligned}$$

[3]In the inequality, all probabilities are conditional on that (G,k) is Yes. We assumed this at the beginning of the proof of (iii).

This completes the proof. □

By repeating $\mathsf{FVS}(G, k)$ 4^k times, we obtain an algorithm summarized in the lemma below.

Lemma 6. *There is an algorithm running in time $O(4^k \cdot poly(n))$ time which, given an input (G, k) to* FEEDBACK VERTEX SET, *outputs*

(i) NO *if (G, k) is a* NO*-instance, and*

(ii) outputs a solution of size at most k with probability $1 - e^{-1}$ if one exists.

Proof: The algorithm $\mathcal{A}$ works as follows: on the input instance (G, k), we run the procedure FVS 4^k times. If a run of FEEDBACK VERTEX SET returns a vertex set S, then return S as an output of $\mathcal{A}$. If all 4^k executions of FVS on (G, k) returns NO, then $\mathcal{A}$ returns NO. Clearly the algorithm $\mathcal{A}$ runs in the claimed running time because FVS runs in polynomial time and $\mathcal{A}$ invokes FVS as a subroutine 4^k times. That $\mathcal{A}$ satisfies (i) follows immediately from Lemma 5. To see (ii), observe that the probability that $\mathcal{A}$ returns NO when (G, k) is YES equals

$$\mathsf{Pr}[\mathsf{FVS}(G,k) \text{ returns NO while } (G,k) \text{ is YES}]^{4^k} \leq (1 - \frac{1}{4^k})^{4^k} \approx e^{-1}(\approx 0.36),$$

where the equality holds because each run of FVS is independent, and the second inequality holds due to Lemma 5. The property (ii) follows. □