\(O^*(5^k)\)-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 \(A\) 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 \(A\) on \((G_{i+1}, X_i \cup \{v_{i+1}\})\). If \(A\) 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 \(A\) 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 \(A\) 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 \( \tilde{X} \subseteq V \), an integer \( k \geq 0 \).

**Question:** Does \( G \) have a feedback vertex set \( \tilde{Y} \) such that \( |\tilde{Y}| \leq k \) and \( \tilde{Y} \cap \tilde{X} = \emptyset \)?

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: \[ \text{procedure dfvs}(G, \tilde{X}, k) \]
2: \[ \quad \text{Let } F = G[V \setminus \tilde{X}] \text{ and } \mathcal{C} \text{ be the set of connected components of } G[\tilde{X}] \]
3: \[ \quad \text{Delete all of degree at most 1. Bypass all degree-2 vertices of } F: \text{ exhaustively.} \]
4: \[ \quad \text{if } G \text{ is acyclic then return } \emptyset. \]
5: \[ \quad \text{else if } G[\tilde{X}] \text{ contains a cycle then return } \text{No}. \]
6: \[ \quad \text{else if } G \text{ contains a cycle and } k = 0 \text{ then return } \text{No}. \]
7: \[ \quad \text{end if} \]
8: \[ \quad \text{Choose a leaf } v \text{ of } F. \quad \text{\(\triangleright k > 0\)} \]
9: \[ \quad \text{if } v \text{ has two neighbors in a single components of } \mathcal{C} \text{ then} \]
10: \[ \quad \quad \text{return dfvs}(G - v, \tilde{X}, k - 1) \cup \{v\} \]
11: \[ \quad \text{else} \quad \text{\(\triangleright v \) has two neighbors belonging to distinct components of } \mathcal{C} \]
12: \[ \quad \quad \text{return dfvs}(G - v, \tilde{X}, k - 1) \text{ or dfvs}(G, \tilde{X} \cup \{v\}, k). \]
13: \[ \quad \text{end if} \]
14: \[ \text{end procedure} \]

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

\(^1\)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.

\(^2\)Creating a connection between the two problems so that by solving instances of the latter, one can obtain a solution to the former.
Proof: We omit the correctness proof (which is rather straightforward, see [1] for details). To analyze the running time, we introduce a measure \( \mu(I) \) of an instance \( I = (G, \tilde{X}, k) \) to \textsc{Disjoint FVS}.

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

In Line 12, each branching decreases the measure \( \mu \) 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.

\[\square\]

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

Proof: Given an instance \( (G, X) \) to \textsc{Compression FVS}, we create an instance \( (G', \tilde{X}, k') \) to \textsc{Disjoint FVS} for every \( I \subseteq X \) 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.

\begin{algorithm}
\caption{Algorithm for \textsc{Compression FVS}}
\begin{algorithmic}[1]
\Procedure{\( \mathcal{B}(G, X) \)}{}
\For{all \( I \subseteq X \)}
\If{\( dfvs(G', \tilde{X}, |\tilde{X}| - 1) \neq \text{No} \)}
\State Let \( \tilde{Y} = dfvs(G', \tilde{X}, |\tilde{X}| - 1) \) and \textbf{return} \( \tilde{Y} \cup I \)
\EndIf
\EndFor
\State \textbf{return} \text{No}
\EndProcedure
\end{algorithmic}
\end{algorithm}

We first observe that if an instance \( (G', \tilde{X}, |\tilde{X}| - 1) \) of \textsc{Disjoint FVS} is a \text{Yes}-instance at Line 3, then the output \( \tilde{Y} \cup I \) is indeed a solution to \( (G, X) \) for \textsc{Compression FVS}. Indeed, \( |\tilde{Y}| \leq |\tilde{X}| - 1 \) implies that \( |\tilde{Y}| + |I| < |\tilde{X}| + |I| = |X| \). 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 \text{No}, then \( (G, X) \) is a \text{No}-instance to \textsc{Compression FVS}.

If \( (G, X) \) is a \text{Yes}-instance to \textsc{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 \textsc{Disjoint FVS}. In particular, there exists some \(I^*\) (not necessarily the same \(I\)) such that the corresponding instance of \textsc{Disjoint FVS} is \textsc{Yes}, and therefore the condition of Line 3 is satisfied. Accordingly, the output of \(B(G, X)\) is not \textsc{No}. This proves the correctness of the algorithm \(B\).

Finally, we observe that for all \(I \subseteq X\) of size \(i\), an instance \(I\) to \textsc{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 \textsc{dfvs} runs in time \(O^*(2^{\mu(I)})\), thus in \(O^*(4^{|X|-i})\) time. Therefore, the algorithm \(B\) runs in time

\[
\sum_{i=0}^{\left\lfloor \frac{|X|}{2} \right\rfloor} \binom{|X|}{i} \cdot 4^{|X|-i} \cdot n^d \leq (4 + 1)^{|X|} \cdot n^d.
\]

\[\square\]

\section*{References}