FPT-approximation for FPT Problems

Hussein Houdrouge

Table of contents

1. Introduction

2. Some Techniques for FPT-Algorithms

   Branching Algorithms

   Iterative Compression

   Important Separators

3. FPT Approximation Algorithms

   Two-extremal Separators Technique

Introduction

Definition
(Parameterised problem [CFK+15]) A parameterized problem is a language $L \subseteq \Sigma^* \times \mathbb{N}$, where Σ is a fixed finite alphabet. For an instance $(x, k) \in \Sigma^* \times \mathbb{N}$, k is called the parameter.

Example: Vertex Cover

Definition (Vertex Cover)
Given a graph $G = (V, E)$, a vertex cover $VC \subseteq V$ such that every edge $e = uv \in E$ has at least one endpoint in VC.

Example: Vertex Cover

Definition (Vertex Cover)
Given a graph $G = (V, E)$, a vertex cover $VC \subseteq V$ such that every edge $e = uv \in E$ has at least one endpoint in VC.

Is there a vertex cover of size k? (G, k) is an instance of the parameterised problem.

FPT Problem and Algorithm

Definition (FPT problem [CFK$^+$15])
A parameterized problem $L \subseteq \Sigma^* \times \mathbb{N}$ is fixed parameter tractable (FPT) if there is an algorithm A (called a fixed parameter algorithm) that decides if an instance $(x, k) \in \Sigma^* \times \mathbb{N}$ is in L in time bounded by $f(k).|(x, k)|^{O(1)}$, where $f : \mathbb{N} \longrightarrow \mathbb{N}$ is a computable function.

FPT Approximation Algorithm

Definition (Optimisation Problem)
An *NP*-optimisation problem is defined as a tuple $(I, sol, cost, goal)$ where

- I is the set of instances.
- For an instance $x \in I$, $sol(x)$ is the set of feasible solutions for x, the length of each $y \in sol(x)$ is polynomially bounded in $|x|$, it can be decided in polynomial time in x whether $y \in sol(x)$ holds for a given x and y.
- Given an instance x and a feasible solution y, $cost(x, y)$ is a polynomial time computable positive integer.
- goal $\in \{\min, \max\}$.

Definition (FPT-Approximation Algorithm)
Let $X = (I, sol, cost, goal)$ be a minimisation problem. A standard factor $c(k)$ FPT-approximation algorithm for X (where the parametrisation is by solution size or value) is an algorithm that, given an input $(\mathbf{x}, \mathbf{k})$ satisfying $\mathbf{opt}(\mathbf{x}) \leq \mathbf{k}$,

Definition (FPT-Approximation Algorithm)
Let $X = (I, sol, cost, goal)$ be a minimisation problem. A standard factor $c(k)$ FPT-approximation algorithm for X (where the parametrisation is by solution size or value) is an algorithm that, given an input $(\mathbf{x}, \mathbf{k})$ satisfying $\mathbf{opt}(\mathbf{x}) \leq \mathbf{k}$,

- It runs in time $f(k) \cdot |x|^{O(1)}$.

Definition (FPT-Approximation Algorithm)
Let $X = (I, sol, cost, goal)$ be a minimisation problem. A standard factor $c(k)$ FPT-approximation algorithm for X (where the parametrisation is by solution size or value) is an algorithm that, given an input $(\mathbf{x}, \mathbf{k})$ satisfying $\mathbf{opt}(\mathbf{x}) \leq \mathbf{k}$,

- It runs in time $f(k) \cdot |x|^{O(1)}$.
- It computes a $y \in sol(x)$ such that $cost(x, y) \leq k.c(k)$.

Definition (FPT-Approximation Algorithm)
Let $X = (I, sol, cost, goal)$ be a minimisation problem. A standard factor $c(k)$ FPT-approximation algorithm for X (where the parametrisation is by solution size or value) is an algorithm that, given an input $(\mathbf{x}, \mathbf{k})$ satisfying $\mathbf{opt}(\mathbf{x}) \leq \mathbf{k}$,

- It runs in time $f(k) \cdot |x|^{O(1)}$.
- It computes a $y \in sol(x)$ such that $cost(x, y) \leq k.c(k)$.
- For inputs not satisfying $opt(x) \leq k$, the output can be arbitrary.

What Problems within FPT?

- Problems with FPT algorithm run in $2^{poly(k)} n^{O(1)}$, where $poly(k)$ is a non linear polynomial in k.
- Problems with lower bounds on $f(k)$ under *ETH* or *SETH*.

Some Techniques for FPT-Algorithms

Branching Algorithms

FPT algorithm for Vertex Cover - (Branching Algorithm)

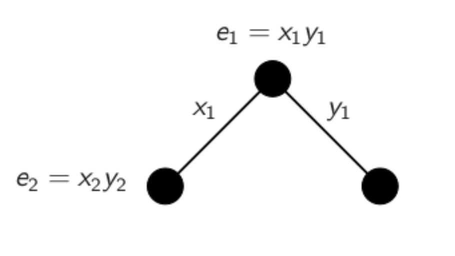


Figure 1: Select an edge, pick a vertex, and delete the incident edges.

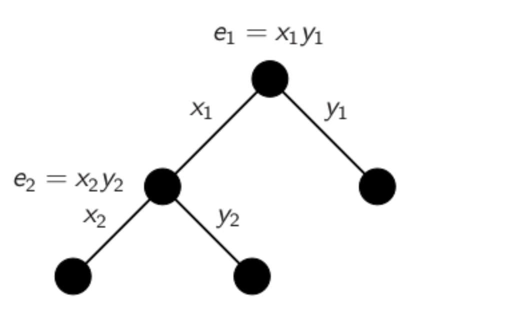


Figure 2: Select a second edge, pick one of its vertices...

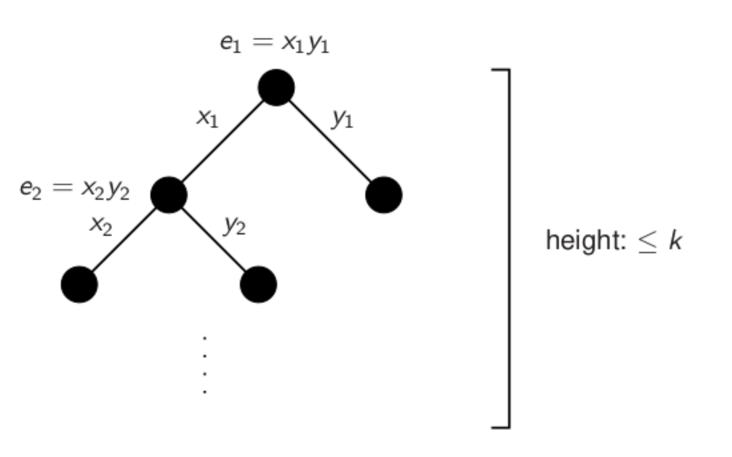
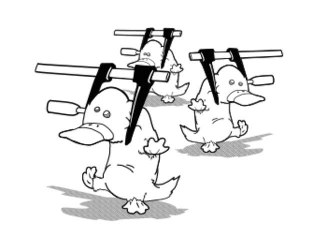


Figure 3: After k, steps in each branch, we have 2^k leaves. At each nodes we performed polynomial time operations which leads to $O(2^k \dot{p}oly(n))$

Iterative Compression



Iterative Compression

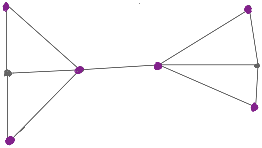


Figure 4: Vertex cover of size at most $2k$, for $k = 4$.

Iterative Compression

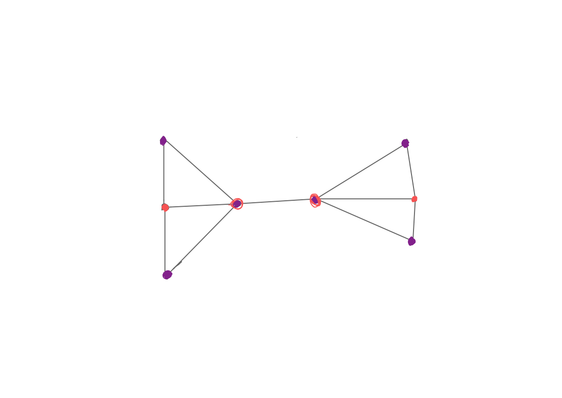


Figure 5: Vertex Cover of size $k = 4$.

Compression Vertex Cover

Input:

- A graph G.
- A solution W of size at most $2k$.
- an integer K.

Output: A solution S of size at most k.

Compression Vertex Cover

Input:

- A graph G.
- A solution W of size at most $2k$.
- an integer K.

Output: A solution S of size at most k. **Note:** if $|W| > 2k$, then it is clear that we do not have a solution.

The Structure of the Solution

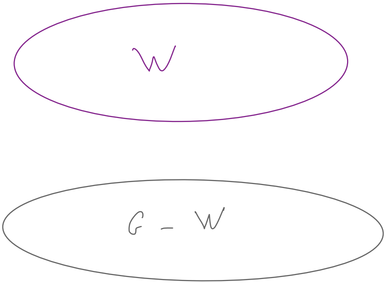


Figure 6: The solution W of size at most $2k$.

Solution-Compression

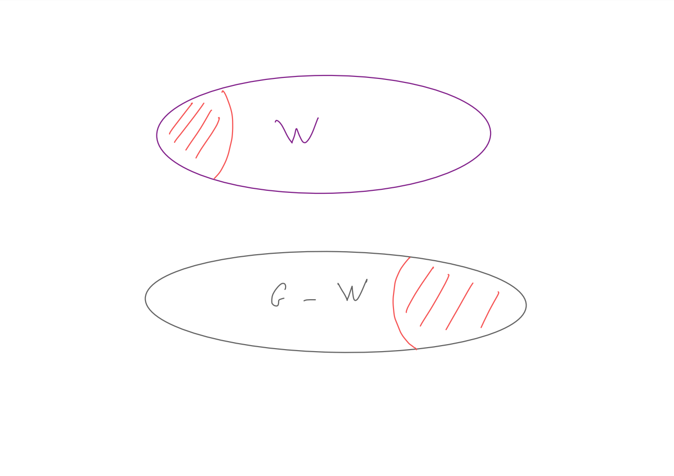


Figure 7: Possible solutions

Disjoint Vertex Cover

Input:

- integer k.
- Solution $W' \subseteq W$ of size $2k - |F|$.
- Graph $G' = G - F$.

Output: A solution S disjoint from W' such that $|F \cup S| \leq k$.

Disjoint-Problem

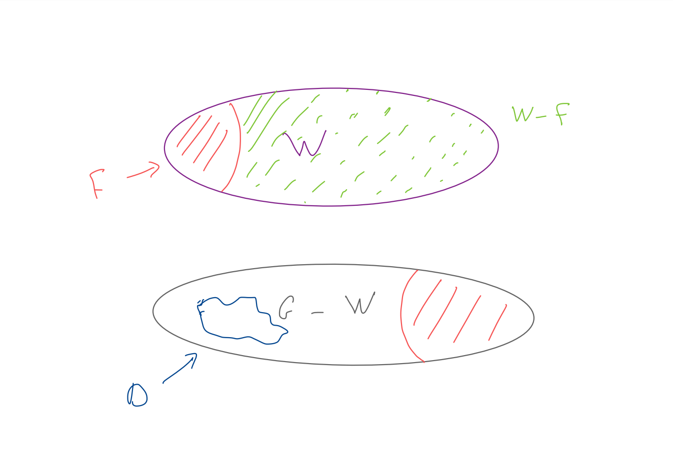


Figure 8: $F \cup D$ possible solution, such that D is disjoing from $W - F$.

Disjoint-Vertex Cover

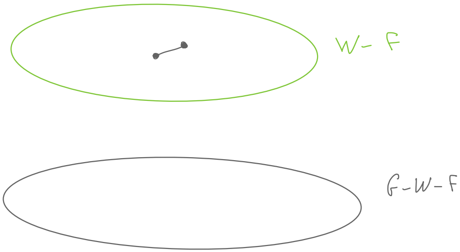


Figure 9: Can $W - F$ contains an edge?

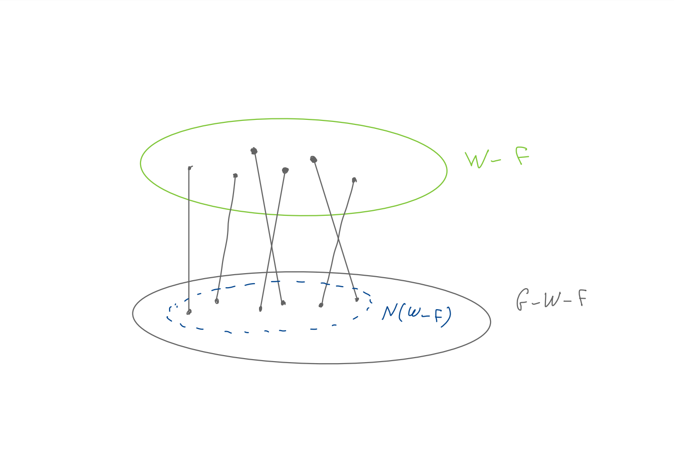


Figure 10: Neighbour of $W - F$ as a solution.

The Compression Algorithm

Algorithm:

1. Branch on all the subset F of W.
2. Solve Disjoint problem on $G - F$ given $W - F$.

Run-time: $2^{|W|} n^{O(1)} \leq 4^k n^{O(1)}$.

The Compression Algorithm

Algorithm:

1. Branch on all the subset F of W.
2. Solve Disjoint problem on $G - F$ given $W - F$.

Run-time: $2^{|W|} n^{O(1)} \leq 4^k n^{O(1)}$.

Can we do better??

The Compression Algorithm

Algorithm:

1. Branch on all the subset F of W.
2. Solve Disjoint problem on $G - F$ given $W - F$.

Run-time: $2^{|W|}n^{O(1)} \leq 4^k n^{O(1)}$.

Can we do better?? Can we get a solution smaller than $2k$?

Iterative Compression

Main Idea:

- Build the instances of the problem iteratively, and apply compression algorithm given a solution of size at most $k + 1$.

Iterative Compression

Consider the vertices of $G : \langle v_1, ..., v_n \rangle$, and let $G_i = G[v_1, ..., v_i]$.

Algorithm:

1. $S = \{v_1\}$
2. For $i = 1$ to $n - 1$.
   2.1 $S = \text{Compress}(G_i, S, k)$
   2.2 if $S > k$ return NO, Otherwise set $S = S \cup \{v_{i+1}\}$ and proceed to the next iteration.

Iterative Compression - Summary

- If there exists an algorithm solving $(*)$-Compression in time $f(k) \cdot n^c$, then there exists an algorithm solving problem $(*)$ in time $O(f(k) \cdot n^{c+1})$.

Iterative Compression - Summary

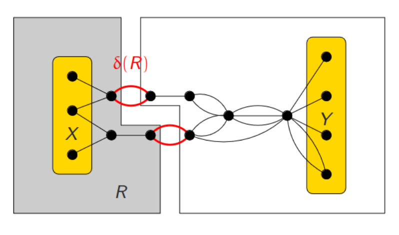
- If there exists an algorithm solving $(*)$-Compression in time $f(k) \cdot n^c$, then there exists an algorithm solving problem $(*)$ in time $O(f(k) \cdot n^{c+1})$.
- If there exists an algorithm solving Disjoint-$(*)$ in time $g(k) \cdot n^{O(1)}$, then there exists an algorithm solving $(*)$-Compression in time

$$\sum_{i=0}^{k} \binom{k+1}{i} g(k-i) n^{O(1)}.$$

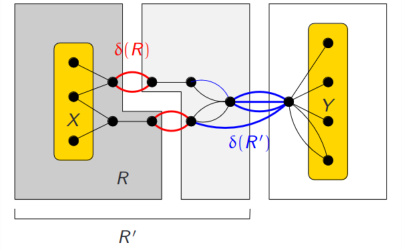
In particular, if $g(k) = \alpha^k$, then (*)-Compression can be solved in time $(1+\alpha)^k \cdot n^{O(1)}$.

Important Separators

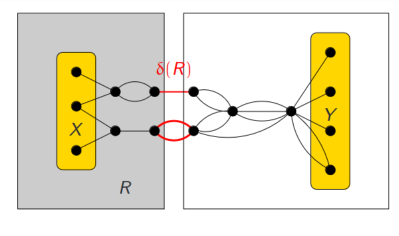
Definition
An $(X, Y)-$separator $\delta(R)$ is important if there is no $(X, Y)-$separator $\delta(R')$ with $R \subset R'$ (Cover) and $|\delta(R')| \leq |\delta(R)|$ (if both condition applies we say $\delta(R')$ dominates $\delta(R)$).



Important Separators



Important Separators



Number of Important Separators

Theorem
There are at most 4^k important (X, Y)-separators of size at most k.

This upper-bound is tight.

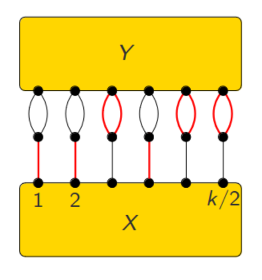


Figure 11: There are exactly $2^k/2$ important (X, Y)-separators of size at most k in this graph.

Number of Important Separators

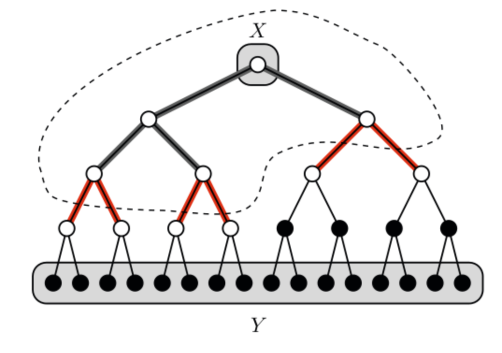


Figure 12: A graph with $\theta(4^k/k^{3/2})$ important (X, Y)-cuts of size k: every full binary sub-tree with k leaves gives rise to an important (X, Y)-cut of size k.

The Multiway CUT

Input:

- A graph $G = (V, E)$.
- A set $T \subseteq V$ of terminals.
- Integer k.

Output: A set $S \subseteq E$ of size at most k such that each connected component of $G - S$ contains at most one $t \in T$.

FPT-Algorithm using Important Separators

Lemma
Let $t \in T$. The Multiway Cut problem has a solution S that contains an important $(t, T - t)$-separator.

FPT-Algorithm using Important Separators

Lemma
Let $t \in T$. The Multiway Cut problem has a solution S that contains an important $(t, T - t)$-separator.

Algorithm (Main idea).

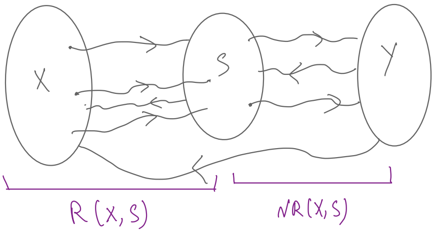
- Choose a t, while t is not alone in a component.
- Enumerate all the important $(t, T - t)$-separators.
- Branch on a separator S of size at most k, set $k = k - |S|$

Important Separators in Digraph

Definition
(Separator in Digraph) A vertex set S disjoint from $X \cup Y$ is called an $X - Y$ separator if there is no $X - Y$ path in $D - S$.

- S is minimal if no strict subset of S is also an $X - Y$-Separator.
- $R_D(X, S)$ the set of vertices reachable from vertices of X via directed path in $D - S$.
- $NR_D(X, S)$ the set of vertices not reachable from vertices of X in $D - S$.

Directed Separators



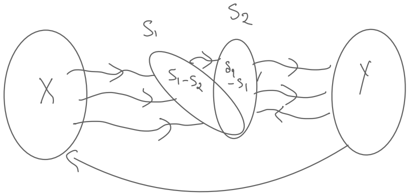
Covering and Dominating

Definition
Let S_1, S_2 be $X - Y$ separators.

- S_2 covers S_1 (denoted by $S_1 \sqsubseteq S_2$) if $R(X, S_1) \subseteq R(X, S_2)$.
- S_2 dominates S_1 (denoted $S_1 \preccurlyeq S_2$) if S_2 covers S_1 and $|S_2| \leq |S_1|$.

Directed Separators

Observation: Let S_1 and S_2 be minimal $X - Y$ separators such that $S_1 \sqsubseteq S_2$. Then, $S_2 - S_1 \subseteq NR(X, S_1)$. Similarly $Y \subseteq NR(S_1 - S_2, S_2)$.



Important Separators in Digraph

Definition
(Important Separators) Let S be a minimal $X - Y$ separator.

- S is important $X - Y$ separator closest to Y, if there is no $X - Y$ separator S' such that $S \preccurlyeq S'$.
- S is important $X - Y$ separator closest to X, if there is no $X - Y$ separator S' such that $S' \preccurlyeq S$.

FPT Approximation Algorithms

Two-extremal Separators Technique

$\mathcal{F}$-transveral problems

Let $\mathcal{F} = \{F_1, F_2, ..., F_q\}$ be a fixed set of sub-graphs of a digraph D such that $\mathcal{F}$-free sub-graphs of D are closed under taking sub-graphs.

Definition
An $\mathcal{F}$-transveral in D is a set of vertices that intersects every $F_i \in \mathcal{F}$.

In the following we are interested in F where every $F_i \in \mathcal{F}$ is strongly connected component.

Goal: Compute the minimum $\mathcal{F}$-transveral set.

Subset Feedback Vertex Set

Definition
Let $D = (V(D), A(D))$ be a directed graph. A directed closed walk in D is an alternating sequence of vertices and arcs that starts and end on the same vertex (it is odd if it has odd number of edges). For a set $T \subseteq V(D) \cup A(D)$,

- A directed closed walk in D is said to be a T-closed walk if it contains an element from T.
- A T-closed walk is called a T-cycle if it is a simple cycle.
- A set $S \subseteq V(D)$ is called a $T - sfvs$ if it is intersects every T-cycle in D.

Taking $\mathcal{F}$ as the set of all T-cycle, implies that SFVS is a special case of Strongly Connected Component (SCC) $\mathcal{F}$-Transveral.

Two-extremal Separator Technique

Lemma
Let $\tilde{S}$ be an $\mathcal{F}$-transveral in D. Let $W = W_1 \uplus W_2$ be an $\mathcal{F}$-transveral in D such that for some $\phi \neq S \subseteq \tilde{S}$, S is a minimal $W_1 - W_2$ separator. Let X_{pre} and X_{post} be $W_1 - W_2$ separator in D such that $X_{pre} \sqsubseteq S \subseteq X_{post}$. Then $\tilde{S} - S$ is an $\mathcal{F}$-transveral in the graph $D' = D - (X_{pre} \cup X_{post})$.

Two-extremal Separator Technique

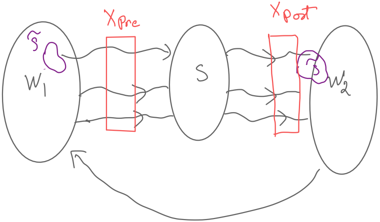


Figure 13: How $\tilde{S}$ looks like.

Two-extremal Separator Technique

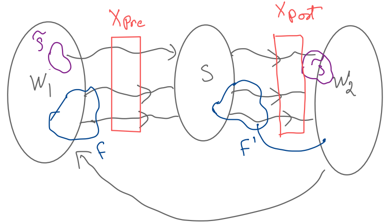


Figure 14: There exists F that is contained in $D'' = D' - (\tilde{S} - S)$

Two-extremal Separator Technique

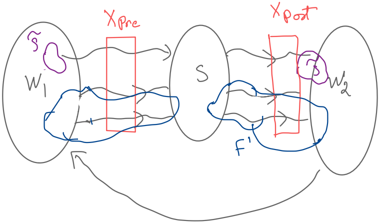


Figure 15: Any F will contains a walk intersecting S and W because both are solutions. But such walk cannot exists because $S - X_{pre}$ is not reachable from W_1 in $D - X_{pre}$...

Two-extremal Separator Technique

Lemma

Let D,W_1,W_2, $\tilde{S}$, S be as defined in the previous lemma. Then, there exists an important $W_1 - W_2$ separator closest to W_1 of size at most $|S|$, call it X_{pre}, and an important $W_1 - W_2$ separator closest to W_2 of size at most $|S|$, call it X_{post}, such that $\tilde{S} - S$ is an important $\mathcal{F}$-transversal in $D' = D - (X_{pre} \cup X_{post})$.

2-Approximation for Subset Directed Feedback Vertex Set

Theorem
There is a factor-2 FPT-approximation algorithm for Subset DFVS with running time $2^{O(k)} n^{O(1)}$.

Strict Subset Directed Feedback Vertex Set

Definition
A factor-c FPT-approximation algorithm for Strict Subset DFVS in an algorithm take the input: (D, T, W, k).

- A directed graph D.
- $T \subseteq V(D) \cup A(D)$.
- $W \subseteq V(D)$ is T-sfvs in D.
- $k \in \mathbb{N}$.

Output:

- A T-sfvs set in time $f(k).n^{O(1)}$ of size at most $c \cdot k$, if there is a T-sfvs S of size at most k, and W is contained in a unique strongly connected component of $D - S$. Otherwise, the output can be arbitrary.

Strict Subset Directed Feedback Vertex Set

Lemma
There is a factor-1 FPT-approximation algorithm for Strict Subset DFVS with running time $2^{O(k)}n^{O(1)}$. We call this algorithm Alg-Strict-SFVS.

Strict Subset Directed Feedback Vertex Set

Proof.

- Let $I = (D, T, W, k)$ be the given input.
- Suppose $(u, v) \in T$ such that $u, v \in W$, then the algorithm terminates with arbitrary output.
- S will break every cycle contains (u, v), $u, v \in W$, will not be in strongly connected component.
- Construct a new tuple $I' = (D', T', w, k)$ where D' is obtained from D by identifying the vertices in W and T' is adjusted accordingly. w is the new vertex created in place of W.

□

Strict Subset Directed Feedback Vertex Set

Lemma
The following statements hold.

1. w *is a* T*-sfvs in* D'
2. *Every* T*-sfvs* S *in* D *that is disjoint from* W *such that* W *is contained in a unique strongly connected component of* $D - S$*, is a* T'*-sfvs in* D' *that is disjoint from* w*.*
3. *Conversely, every* T'*-sfvs in* D' *disjoint from* w *is a* T*-sfvs in* D*.*

Therefore, it is sufficient to give an algorithm for I'.

Strict Subset Directed Feedback Vertex Set

Lemma
Let S be a solution for I'. For every $(u, v) \in T'$, either $\{u, v\} \cap S \neq \phi$ or there is a solution for I' that contains an important $x - w$ separator closest to w for some $x \in \{u, v\}$.

Strict Subset Directed Feedback Vertex Set

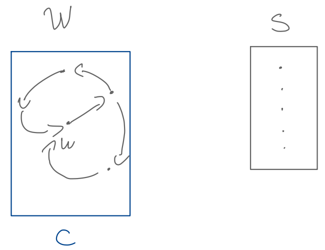


Figure 16: S is a solution, W contained in SCC C.

Strict Subset Directed Feedback Vertex Set

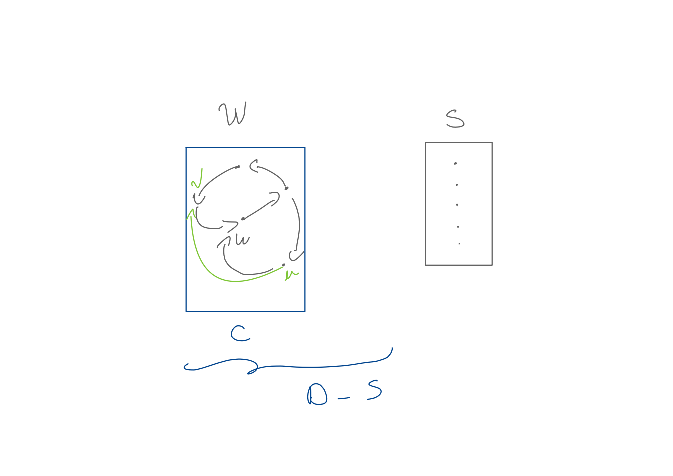


Figure 17: C cannot contain an edge of T' because it contradict that S breaks every cycle containing T.

Strict Subset Directed Feedback Vertex Set

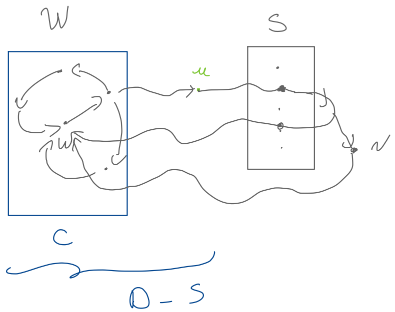


Figure 18: S either contains u or v, or intersects every $u - w$, $v - w$ path.

Strict Subset Directed Feedback Vertex Set

It remains to show if S intersects $w - x$ or $x - w$ paths for some $x \in \{u, v\}$, then there is a solution S' that contains an important $w - x$ separator closest to w for some $x \in \{u, v\}$.

Strict Subset Directed Feedback Vertex Set

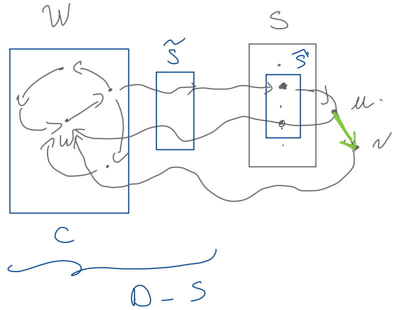


Figure 19: $\hat{S} \subset S$ is a minimal $w - x$ separator, $\tilde{S}$ is an important separator closest to w of size $|\hat{S}|$ covered by $\hat{S}$.

Strict Subset Directed Feedback Vertex Set

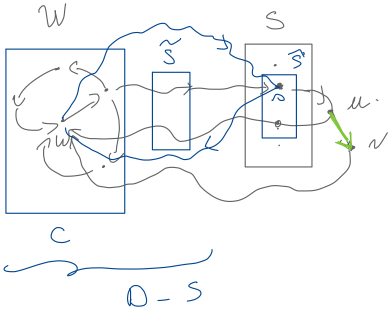


Figure 20: $(S - \hat{S}) \cup \tilde{S}$ is also a solution for I'. If there is a closed walk through w and s, it contradicts our observation.

Algorithm For Strict SDFVS

Algorithm: for an arc $e = (u, v) \in T$

- Branch 1: add u to S, set $k = k - 1$.
- Branch 2: add v to S ...
- Branch 3: Enumerate all important $u - w$ separator closest to w of size at most k.
- Branch 4: Enumerate all $v - w$ separators closest to w of size at most k.
- Branch 5: Enumerate all $w - u$ closest to w of size at most k.
- Branch 6: Enumerate all $w - v$ closest to w of size at most k.

Run-time: $(10 + 4\sqrt{6})^k n^{O(1)}$.

Algorithm SDFVS

Lemma
Let D be a digraph, $T \subseteq A(D)$, and let W and S be disjoint T-sfvs in D. Let $\phi \neq W' \subseteq W$ be such that in $D - S$, there is a strongly connected component whose intersection with W is precisely W'. Consider the graph D' obtained from D by adding a bi-directed clique on W' (i.e., we add an arc (w, w') for every $w, w' \in W'$ such that $(w, w') \notin A(D)$). Then, W and S are both T-sfvs in D'.

A factor-2 approximation algorithm for Subset DFVS

Algorithm:

1. $V(D) = \{v_1, ..., v_n\}$, $V_i = \cup_{j=1}^{i} v_j$ for $i \in [n]$.
2. For $X \subseteq V(D)$, let $T[X] = \{(x, y) \in T | x, y \in X\}$.
3. $W_1 = \{v_1\}$.
4. For $i = 1$ to n:
   - 4.1 $I_i = (D[V_i], T[V_i], W_i, k)$
   - 4.2 $S =$ Alg-Compression-SFVS(I_i).
   - 4.3 $W_{i+1} := \{v_{i+1}\} \cup S$

A factor-2 approximation algorithm for Subset DFVS

Alg-Compression-SFVS:

- Input: (D, T, W, k)
- Output: if there is a T-sfvs S in D of size at most k that is not necessarily disjoint from W then it outputs a T-sfvs in D of size at most $2k$ in $2^{O(k+|w|)}n^{O(1)}$.

Alg-Disjoint-SFVS:

- Input: (D, T, W, k)
- Output: if there is a T-sfvs S in D of size at most k that is disjoint from W, then it outputs a T-sfvs in D of size at most $2k$ in $2^{O(k+|w|)}n^{O(1)}$.

Alg-Disjoint-SFVS

Base Case: $k \leq 1$ or $|W| = 1$, we can solve the problem by Brute force.

- If $k \leq 1$, it is sufficient to check whether there is a T-cycle in D and if yes, whether there is a T-sfvs in D of size at most 1.
- if $k > 1$, $|W| = 1$, then we can simply return W.

Alg-Disjoint-SFVS

Definition
$rel_D(X, Y)$ the set of all vertices that lie in a strongly connected component of $D - Y$ intersected by X.

Definition
Let $\mathcal{P}$ be the set of all 3-partitions of W into (X, Y, Z). For every $\tau = (X, Y, Z) \in \mathcal{P}$, we define the following tuples. Let $1 \leq i, j \leq k$.

- $\mathcal{L}^i[Z \longrightarrow XY]$ denotes the set of all important $Z - X \cup Y$ separators of size at most i closest $X \cup Y$.
- $\mathcal{L}^i[XY \longleftarrow Z]$ denotes the set of all important $Z - X \cup Y$ separators of size at most i closest to Z.

Alg-Disjoint-SFVS

Definition
For $1 \leq i, j \leq k$, let $L_1 \in \mathcal{L}^j[Z \longrightarrow XY]$ and $L_2 \in \mathcal{L}^j[XY \longleftarrow Z]$.

- $\mathcal{L}^i[Y \longrightarrow X, L_1, L_2]$ denotes the set of all important $Y - X$ separators of size at most i closest to X in $D - (L_1 \cup L_2)$.
- $\mathcal{L}^i[X \longleftarrow Y, L_1, L_2]$ denotes the set of all important $Y - X$ separators of size at most i closest to Y in $D - (L_1 \cup L_2)$.

Alg-Disjoint-SFVS

For $Q \subseteq V(D)$:

- $I[Q, Z, i]$ denotes the tuple $(D[\mathit{rel}(Z, Q)], T, Z, i)$.
- $I[Q, XY, i]$ denotes $(D[\mathit{rel}(X \cup Y), Q], T, X \cup Y, i)$.
- $I[Q, X, i]$ denotes $(D[\mathit{rel}(X, Q)], T, X, i)$.
- $\tilde{I}[Q, Y, i]$ denotes (D', T, Y, i), where D' is the graph obtained from $D[\mathit{rel}(Y, Q)]$ by adding a bidirected clique on Y.

Alg-Disjoint-SFVS

Algorithm:

1. For every $(X, Y, Z) \in \mathcal{P}$ such that:
   - $|W|/3 \leq |X \cup Y|$ and $|Z| \leq 2|W|/3$.
   - $|Y| > |W|/3$.

1.1 For every $1 \leq i_1 \leq k$, and for every i_2 such that $k_1 = i_1 + i_2 \leq k$:

- **Step 1** Guess:
  - $L_1 \in \mathcal{L}^{i_1}[Z \longrightarrow XY]$.
  - $L_2 \in \mathcal{L}^{i_1}[XY \longleftarrow Z]$.
  - $L_3 \in \mathcal{L}^{i_2}[Y \longrightarrow X, L_1, L_2]$.
  - $L_4 \in \mathcal{L}^{i_2}[X \longleftarrow Y, L_1, L_2]$.
  - Set $Q = \cup_{q \in [4]} L_q$

Alg-Disjoint-SFVS

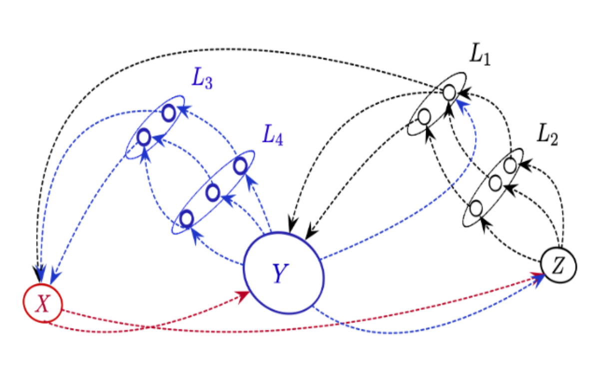


Figure 21: An illustration of the sets $X \uplus Y \uplus Z = W$ and the separators $L_1, \ldots, L_4$. The dotted arrows represents the paths.

Alg-Disjoint-SFVS

- **Step 2:** If $|W|/3 \leq |X \cup Y|, |Z| \leq 2|W|/3$, then:
  For every $i_3 + i_4 \leq k - k_1$:
  1. $S_z =$ Alg-Dijoint-SFVS($I[Q, Z, i_3]$)
  2. $S_{XY} =$ Alg-Disjoint-SFVS($I[Q, XY, i_4]$).
  3. $\Delta = Q \cup S_Z \cup S_{XY}$ is a T-sfvs in D of size at most $2k$, then we return Δ.
- **Step 3:** if step 2 does not apply and $Y > |W|/3$, then:
  for every i_3, i_4, i_5 such that $i_3 + i_4 + i_5 = k - k_1$
  1. $S_z =$ Alg-Dijoint-SFVS($I[Q, Z, i_3]$).
  2. $S_X =$ Alg-Dijoint-SFVS($I[Q, X, i_4]$).
  3. $S_Y =$ Alg-Strict-SFVS($\tilde{I}[Q, Y, i_5]$).
  4. if $\Delta = Q \cup S_Z \cup S_X \cup S_Y$ is a T-sfvs in D of size at most $2k$, then return Δ.

Proof of Correctness

- Induction on $|W|$.
- The base case $|W| = 1$, in which case, the algorithm works on brute force and hence it is correct.
- Assume $|W| > 1$, suppose there is a T-sfvs S of size at most k in D Disjoint from W.
- We aim to show that the algorithm output a T-sfvs of size at most $2k$.

Proof of Correctness

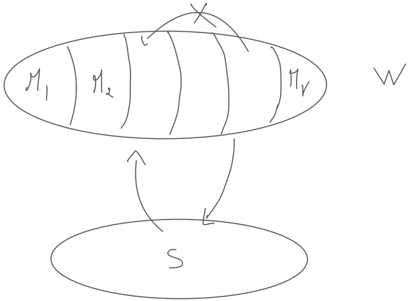


Figure 22: How *W* could Break?

Proof of Correctness

Case 1: $|M_\ell| \leq \frac{|W|}{3}$ for every $\ell \in [r]$.

- Let $\ell' \in [r]$ denote the least value such that $|w|/3 < \sum_{i=1}^{\ell'} |M_i|$. Then,

$$\frac{|W|}{3} < \sum_{i=1}^{\ell'} |M_i| \leq \frac{2|W|}{3}.$$

- Define $X = \phi$, $Y = \cup_{i=1}^{\ell'} M_i$, and $Z = \cup_{\ell'+1}^{r} M_i$.
- Therefore, we have

$$\frac{|W|}{3} \leq |X \cup Y|, |Z| \leq \frac{2|W|}{3}.$$

Then, when considering the partition (X, Y, Z), **step 2** would have been executed.

Proof of Correctness

- Let S_1 be a minimal subset of S that intersects all the $Z - X \cup Y$ paths in D. Let $i_1 = |S_1|$.
- Since D is strongly connected component it follows that $i_1 > 0$ by the two extremal separator lemma, $L_1 \in \mathcal{L}^{i_1}[Z \longrightarrow XY]$ and $L_2 \in \mathcal{L}^{i_2}[XY \longrightarrow Z]$ such that $S' = S - S_1$ is a T-sfvs in $D - (L_1 \cup L_2)$. Let $i_2 = 0$ that implies $L_3 = L_4 = \phi$.
- Let $Q = \cup_{q \in [4]} L_q$, define:
  - $S'_z = S' \cap rel(Z, Q)$, $i_3 = |S'_Z|$.
  - $S'_{XY} = S' \cap rel(X \cup Y, Q), i_4 = |S'_{XY}|$.
- By induction hypothesis we get a solution of size at most $2i_3 + 2i_4$.
- Combining all solutions we get one of size at most $2i_1 + 2i_3 + 2i_4 \leq 2k$.

Proof of Correctness

Case 2: There is $\ell^* \in [r]$ such that $|M_{\ell^*}| > \frac{|W|}{3}$.

- Define $Y = M_{\ell^*}$.
- Similar reasoning as before.
- Exercise for the listener...

References

- M. Cygan, F.V. Fomin, Ł. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh.
  Parameterized Algorithms.
  Springer International Publishing, 2015.
- Daniel Lokshtanov, Pranabendu Misra, M. S. Ramanujan, Saket Saurabh, and Meirav Zehavi.
  FPT-approximation for FPT Problems, pages 199–218.

Thank You!