Contraction to Generalization of Trees

A. Agrawal[1] S. Saurabh[1,2] and P. Tale[2]

September 7, 2017

[1] University of Bergen, Bergen, Norway

[2] The Institute of Mathematical Sciences, HBNI, Chennai, India

Outline

Graph Contraction Problems

Problem Definition

FPT Algorithm

No-Polynomial Kernel

Lossy Kernelization

Graph Contraction Problems

Graph Contraction Problems

$\mathcal{F}$ is a graph class and G/F is graph obtained from G by contracting edges in F

$\mathcal{F}$-Contraction	**Parameter:** k

Input: A graph G and an integer k
Question: Does there exist $F \subseteq E(G)$ of size at most k such that G/F is in $\mathcal{F}$?

$\mathcal{F}$-Contraction: Parameterized Complexity

[HvtHL+12]	Tree Contraction Path Contraction	4^k $2^{k+o(k)}$
[GvtHP13]	Planar Contraction	FPT
[CG13]	Clique Contraction	$2^{\mathcal{O}(k \log k)}$
[HvtHLP13] [GM13]	Bipartite Contraction	FPT $2^{\mathcal{O}(k^2)}$

$\mathcal{F}$-Contraction: Parameterized Complexity

Theorem

$\mathcal{F}$-Edge Contraction *is* $W[2]$*-hard if*

- *[LMS13] [CG13]* $\mathcal{F}$ *can be characterized as* $P_{\ell+1}$*-free graphs or* C_ℓ*-free graphs for* $\ell \geq 4$.

  P_ℓ *and* C_ℓ *are path and cycle on* ℓ *vertices, respectively.*
- *[ALSZ17]* $\mathcal{F}$ *is Split Graphs*

$\mathcal{F}$-Contraction: Kernelization

Theorem

[HvtHL$^+$12] Tree Contraction *does not admit a polynomial kernel unless NP $\subseteq$ coNP/poly and* Path Contraction *admits a linear vertex kernel.*

Starting Point

1. Why is there a polynomial kernel for $\textsc{Path}$s but not for $\textsc{Tree}$s?
2. Why is $\mathcal{F}$-$\textsc{Contraction}$ FPT when $\mathcal{F}$ is $\textsc{Tree}$s (which are C_3-free) but $W[1]$-hard when $\mathcal{F}$ is family of C_t-free graphs ($t \geq 4$)? Or other simple graph classes like P_{t+1}-free graphs Or Split Graphs?

Starting Point

What additional parameter we can associate with $\mathcal{F}$-CONTRACTION such that :

1. it admits a polynomial kernel?
2. an FPT algorithm for super classes of TREEs?

Starting Point

What additional parameter we can associate with $\mathcal{F}$-CONTRACTION such that :

1. it admits a polynomial kernel? Paths to Trees (CIAC '17)
2. an FPT algorithm for super classes of TREEs? This Work

Problem Definition

Generalization of Tree Contraction

$\mathcal{F}_\ell := \{T \mid T$ can be made into a tree by deleting at most ℓ edges$\}$

Observe that $\mathcal{F}_0$ is family of TREES

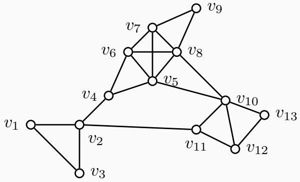
$\mathcal{F}_\ell$-CONTRACTION	**Parameter:** k
Input: A graph G and an integer k.	
Question: Does there exist $F \subseteq E(G)$ of size at most k such that $G/F \in \mathcal{F}_\ell$?	

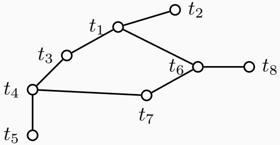
Results:

- FPT algorithm running in time $\mathcal{O}((2\sqrt{\ell}+2)^{\mathcal{O}(k+\ell)} \cdot n^{\mathcal{O}(1)})$.
- No polynomial kernel, when parameterized by k, for any (fixed) $\ell \in \mathbb{N}$.
- Lossy Kernelization

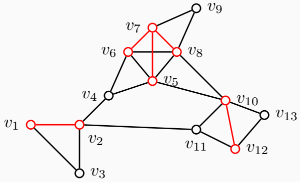
Contraction as a Partition Problem

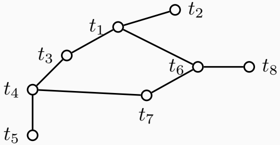
$\mathcal{F}$-Contraction as a Partition Problem



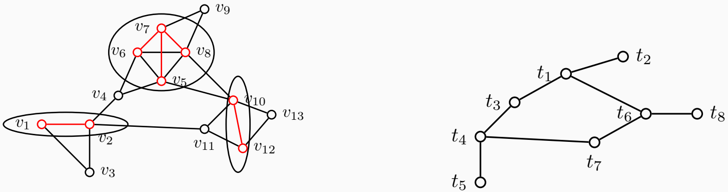


$\mathcal{F}$-Contraction as a Partition Problem





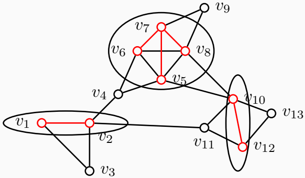
$\mathcal{F}$-Contraction as a Partition Problem

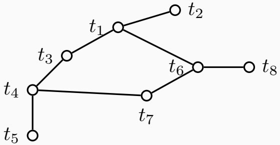


G is contractible to T if there exists a partition of $V(G)$ into $W(t_1), W(t_2), \ldots W(t_{|V(T)|})$ s.t.

- $\forall\ t \in V(T)$, $G[W(t)]$ is connected
- $t_i t_j \in E(T)$ iff $W(t_i)$ and $W(t_j)$ are adjacent in G

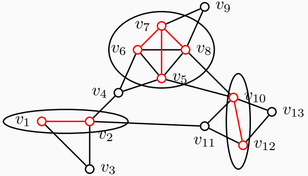
Witness Structure : Definition

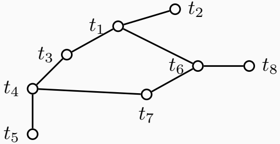




- $\mathcal{W} = \{W(t) \mid t \in V(T)\}$ is called the *T-witness structure* of G
- *Big-witness set* if $|W(t)| > 1$ e.g. $W(t_1), W(t_6), W(t_4)$
- $k = \sum_{t \in V(T)}(|W(t)| - 1)$
  We say G is k-contractible to graph T

Witness Structure : Observations



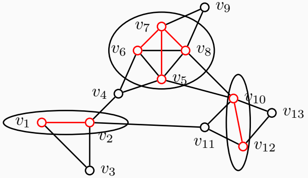


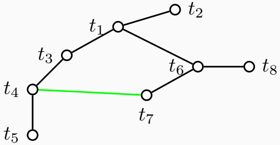
If G is k-contractible to T and $\mathcal{W}$ be its T-witness structure then,

- No witness set in $\mathcal{W}$ contains more than $k + 1$ vertices;
- $\mathcal{W}$ has at most k big witness sets;
- Union of big witness sets in $\mathcal{W}$ contains at most $2k$ vertices.

FPT Algorithm

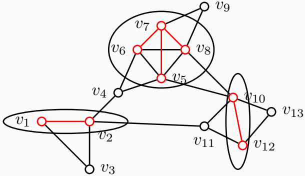
Few Definitions

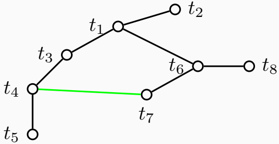




$\mathcal{W} \leftarrow$ a T-witness structure of G

Fix a spanning tree of T and mark edges outside the spanning tree

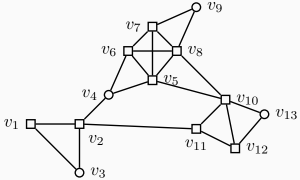


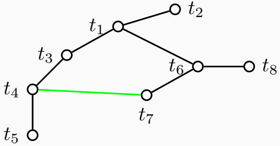


Important Nodes in T

- Nodes corresponding to *big-witness* set (t_1, t_4, t_6)
- Nodes incident on extra edges (t_4, t_7)
- Nodes of degree at least 3 (t_1)

Few Definitions

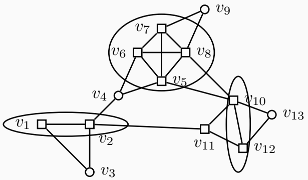


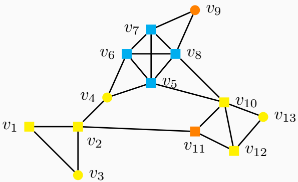


Important vertices in G are the vertices contained in bags corresponding to important nodes.

Lemma

There are at most $\mathcal{O}(k + \ell)$ important vertices.

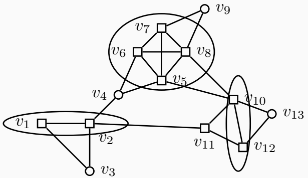


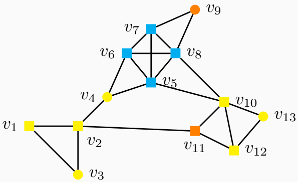


Good Coloring $\phi : V(G) \to [2\sqrt{\ell} + 2]$ is good coloring (wrt $\mathcal{W}$) if

1. Each witness set is monochromatic (Ex. $\{v_5, v_6, v_7, v_8\}$)
2. Color of two *important vertices* which are adjacent or connected by path consisting of *non-important vertices* are different. (Ex. v_2, v_{11})

Randomized FPT Algorithm





Step 1: Color vertices of input graph uniformly at random with $\sqrt{\ell} + 2$ colors.

Step 2: Extract witness sets out of each colored components of a *good* coloring.

Ex. Extract $\{v_1, v_2\}$ out of $\{v_1, v_2, v_3, v_4\}$

Randomized FPT Algorithm

Step 1: Color vertices of input graph uniformly at random with $\sqrt{\ell}+2$ colors.

Lemma

Probability that a random coloring is a good coloring is sufficiently high.

Step 2: Extract witness sets out of each colored components of a *good* coloring.

Lemma

Extracting a witness set from a color class is equivalent of finding its connected vertex cover.

Theorem

$\mathcal{F}_\ell$-Contraction *is* FPT *when parameterized by* k *for fixed* ℓ.

No-Polynomial Kernel

No-Polynomial Kernel

Theorem ([HvtHL+12])

Tree Contraction *does not admit a polynomial kernel unless* $NP \subseteq coNP/poly$.

Lemma

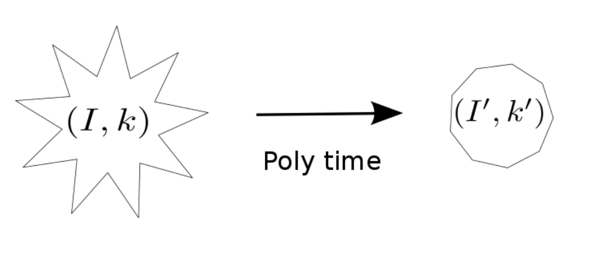
(G, k) *is a* YES *instance of* Tree Contraction *if and only if* (G', k') *is a* YES *instance of* $\mathcal{F}_\ell$-Contraction, *here* $k' = k$.

Theorem

$\mathcal{F}_\ell$-Contraction *does not admit a polynomial kernel unless* $NP \subseteq coNP/poly$.

Lossy Kernelization

Kernelization



Parameterized problem Q admits a $h(k)$-kernel if there exists a poly-time algorithm $\mathcal{A}$ which given an input (I, k) outputs (I', k') such that

- $|I'| + k' \leq h(k)$
- (I, k) is YES instance iff (I', k') is YES instance

How about optimization version?

Optimization Version

For a parameterized problem Q, its optimization analogue is a computable function

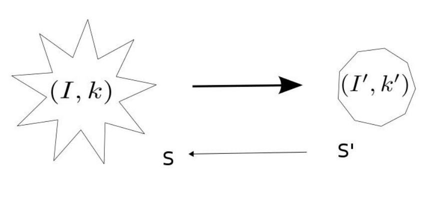
$$\Pi : \Sigma^* \times \mathbb{N} \times \Sigma^* \to \mathbb{R} \cup \{\pm\infty\}$$

Given instance I, parameter k and a solution S, the *value* of a solution S to an instance (I, k) of Q is $\Pi(I, k, S)$.

For parameterized minimization problems,

$$\mathrm{OPT}_{\Pi}(I, k) = \min_{S \in \Sigma^*} \{\Pi(I, k, S)\}$$

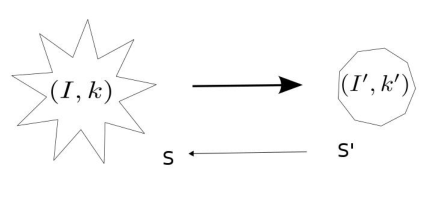
Lossy Kernelization



Given a solution S' of (I', k') can we construct a solution S of (I, k) which is as good as S'?

Quality of solution S' of (I', k') is $\frac{\Pi(I',k',S')}{\mathrm{OPT}(I',k')}$

Lossy Kernelization



Given (I', k', S') can we construct a solution S of (I, k) such that

$$\frac{\Pi(I, k, S)}{\mathrm{OPT}(I, k)} \leq \alpha \frac{\Pi(I', k', S')}{\mathrm{OPT}(I', k')}$$

for some constant α?

Definition (α-PTAS)

An α-*approximate polynomial-time preprocessing algorithm* (α-PTAS) is pair of two polynomial time algorithms as follows:

	Input	Output
Reduction Algorithm	(I, k)	(I', k')
Solution Lifting Algorithm	(I, k) and (I', k', S')	S

such that

$$\frac{\Pi(I, k, S)}{\mathrm{OPT}(I, k)} \leq \alpha \cdot \frac{\Pi(I', k', S')}{\mathrm{OPT}(I', k')}$$

Lossy Kernelization

Definition (α-approximate kernel)

For a parameterized minimization problem Π if

1. α-PTAS
2. the size of the output instance is upper bounded by a computable function $g : \mathbb{N} \to \mathbb{N}$ of k.

Strict α-approximate kernel

Definition (Strict α-approximate kernel)

An α-approximate kernel is said to be *strict* if the solution lifting algorithm returns a solution S such that

$$\frac{\Pi(I, k, S)}{\mathrm{OPT}(I, k)} \leq \max\{\frac{\Pi(I', k', S')}{\mathrm{OPT}(I', k')}, \alpha\}$$

Lossy Kernelization

$$\Pi(I, k, S) = \begin{cases} \infty & \text{if } \ S \ \text{ is not a solution} \\ \min\{|S|, k+1\} & \text{otherwise} \end{cases}$$

Since we are interested in solutions of size at most k

Lossy Kernelization : Reduction Rule

Definition (α-safe)

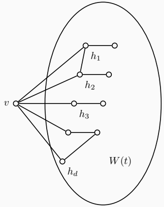
A reduction rule is α-*safe* for Π if there is a solution lifting algorithm s.t. together they constitute a strict α-approximate polynomial-time preprocessing algorithm for Π.

Lossy Kernelization

Let (G, k) be an instance of $\mathcal{F}_\ell$-Contraction and $\alpha > 1$ is a fixed constant. Fix $d = \lceil \frac{\alpha}{\alpha - 1} \rceil$

In *polynomial time* ($n^{\mathcal{O}(d)}$) we can find vertices $h_1, h_2, \ldots, h_d$ s.t.

- all these vertices are in witness set $W(t)$
- there exists v such that $\{h_1, h_2, \ldots, h_d\} \subseteq N(v)$

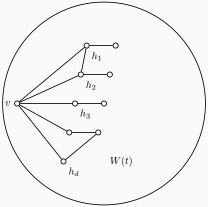


Can we utilize this information to simplify graph?

Lossy Kernelization

Notice : we can't find entire $W(t)$; v may or may not be in $W(t)$.

Introducing Lossy-ness : Add vertex v to $W(t)$

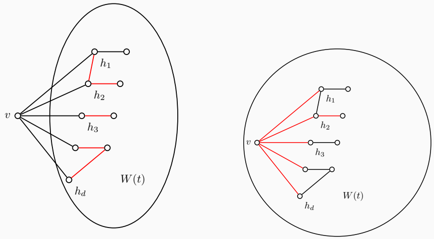


Contract all edges vh_i for all $i \in [d]$ to get new instance $(G', k - (d-1))$

Notice : We contracted d edges but reduced the budget by $d-1$.

Lossy Kernelization

$h_1, h_2, \ldots, h_d$ are in big-witness set $\Rightarrow$ there are $d-1$ solution edges incident these vertices



We are contracting d-many edges for every $(d-1)$ edges in the solution.
The number of extra edge contracted in this process is $\frac{d}{d-1} = \alpha$ factor of the optimum solution

Lossy Kernelization

This reduction rule coupled with other two reduction rules leads to following theorem.

Theorem

$\mathcal{F}_\ell$-Contraction *admits a strict PSAKS, where the number of vertices is bounded by* $c[k(k+2\ell)]^{(\lceil \frac{\alpha}{\alpha-1} \rceil + 1)}$*, where* c *is some fixed constant.*

Thank you!

A. Agrawal, D. Lokshtanov, S. Saurabh, and M. Zehavi.
Split contraction: The untold story.
In *34th Symposium on Theoretical Aspects of Computer Science, STACS 2017, Hannover, Germany*, pages 5:1–5:14, 2017.

Leizhen Cai and Chengwei Guo.
Contracting few edges to remove forbidden induced subgraphs.
In *IPEC*, pages 97–109, 2013.

References ii

Sylvain Guillemot and DÃąniel Marx.
A faster FPT algorithm for bipartite contraction.
Inf. Process. Lett., 113(22–24):906–912, 2013.

Petr A. Golovach, Pim van ’t Hof, and Daniel Paulusma.
Obtaining planarity by contracting few edges.
Theoretical Computer Science, 476:38–46, 2013.

Pinar Heggernes, Pim van ’t Hof, Benjamin Lévêque, Daniel Lokshtanov, and Christophe Paul.
Contracting graphs to paths and trees.
In *Proceedings of the 6th International Conference on Parameterized and Exact Computation*, IPEC’11, pages 55–66, Berlin, Heidelberg, 2012. Springer-Verlag.

References iii

Pinar Heggernes, Pim van 't Hof, Daniel Lokshtanov, and Christophe Paul.
Obtaining a bipartite graph by contracting few edges.
SIAM Journal on Discrete Mathematics, 27(4):2143–2156, 2013.

Daniel Lokshtanov, Neeldhara Misra, and Saket Saurabh.
On the hardness of eliminating small induced subgraphs by contracting edges.
In *IPEC*, pages 243–254, 2013.