Lossy Kernels
for Graph Contraction Problems

Prafullkumar Tale

The Institute Of Mathematical Sciences, Chennai, India

Graph Modification Problems

$\mathcal{H}$ - Modification Problems

Input : Graph G, integer k **Para** : k

Output : Can we make at most k modifications in G so that resulting graph is in $\mathcal{H}$?

Graph Modification Problems

$\mathcal{H}$ - Modification Problems

Input : Graph G, integer k

Para : k

Output : Can we make at most k modifications in G so that resulting graph is in $\mathcal{H}$?

$\mathcal{H}$ — Polynomial time recognisable graph class

Graph Modification Problems

$\mathcal{H}$ - Modification Problems

Input : Graph G, integer k

Para : k

Output : Can we make at most k modifications in G so that resulting graph is in $\mathcal{H}$?

$\mathcal{H}$ — Polynomial time recognisable graph class

Modification — (1) vertex deletion (2) edge deletion
(3) edge addition (4) edge contraction

$\mathcal{H}$- Modification Problems $\equiv$ $\mathcal{F}$-Free -Modification Problems

Modification — (1) vertex deletion (2) edge deletion
(3) edge addition (4) edge contraction

$\mathcal{H}$- Modification Problems $\equiv$ $\mathcal{F}$-Free -Modification Problems

Modification — (1) vertex deletion (2) edge deletion (3) edge addition (4) edge contraction

Problem	$\mathcal{H}$	$\mathcal{F}$ -Free	Modification
VERTEX COVER	Empty Graphs	P_2	Vertex Deletion
EDGE BIPARTIZATION	Bipartite Graphs	$C_3, C_5, C_6, \ldots$	Edge Deletion
MINIMUM FILL-IN	Chordal Graphs	$C_4, C_5, C_6, \ldots$	Edge Addition
CLUSTER EDITING	Set of Cliques	P_3	Edge Add + Del.

Modification — (1) vertex deletion (2) edge deletion
(3) edge addition (4) edge contraction

Theorem (Cai 1996): If $\mathcal{F}$ is bounded and (1), (2), and/or (3) then $\mathcal{F}$-Free -Modification Problems are FPT.

- VERTEX COVER is FPT
- CLUSTER VERTEX EDITING is FPT.

Modification — (1) vertex deletion (2) edge deletion
(3) edge addition (4) edge contraction

If $\mathscr{F}$ is infinite and (1), (2), and/or (3) then

$\mathscr{F}$-Free -Modification Problems are FPT.

- MINIMUM FILL-IN is FPT (Cai '96 + Kaplan '96).
- EDGE BIPARTIZATION is FPT (S. Wernicke 2003).

Modification — (1) vertex deletion (2) edge deletion
(3) edge addition (4) edge contraction

$\mathcal{F}$-Free -Modification Problems when only (4) is allowed?

Modification — (1) vertex deletion (2) edge deletion
(3) edge addition (4) edge contraction

$\mathcal{F}$-Free -Modification Problems when only (4) is allowed?

TREE CONTRACTION	4^k	No poly kernel	Heggernes et al. (2011)
CLIQUE CONTRACTION	exp(k log (k))	No poly kernel	Cai et al. (2013)
PLANAR CONTRACTION	FPT	—	Golavach et al. (2013)
BIPARTITE CONTRACTION	exp(k^2)	—	Guillemot + Mark (2013)

Modification — (1) vertex deletion (2) edge deletion
(3) edge addition (4) edge contraction

$\mathcal{F}$-Free -Modification Problems when only (4) is allowed?

Can we have generic theorem as that of Cai 1996?

Modification — (1) vertex deletion (2) edge deletion
(3) edge addition (4) edge contraction

$\mathcal{F}$-Free -Modification Problems when only (4) is allowed?

Can we have generic theorem as that of Cai 1996? NO

Modification — (1) vertex deletion (2) edge deletion
(3) edge addition (4) edge contraction

$\mathcal{F}$-Free -Modification Problems when only (4) is allowed?

Can we have generic theorem as that of Cai 1996? NO

Theorem (Lokshtanov et al. 2013 + Cai et al. 2013):
If $\mathcal{F}$ contains only one graph (a path on >=5 or cycle on >= 4 vertices) then $\mathcal{F}$-Free-Modification Problems are is W[2]-hard.

Theorem (Lokshtanov et al. 2013 + Cai et al. 2013):
If $\mathcal{F}$ contains only one graph (a path on >=5 or cycle on >= 4 vertices) then $\mathcal{F}$-Free-Modification Problems are is W[2]-hard.

CHORDAL CONTRACTION is W[2]-hard.

Theorem (Lokshtanov et al. 2013 + Cai et al. 2013):
If $\mathcal{F}$ contains only one graph (a path on >=5 or cycle on >= 4 vertices) then
$\mathcal{F}$-Free-Modification Problems are is W[2]-hard.

CHORDAL CONTRACTION is W[2]-hard.

Theorem (Agrawal et al. 2017): SPLIT CONTRACTION is W[1]-hard.

$\mathcal{F}$-Free-Modification Problems are more difficult than their counterparts.

$\mathcal{F}$-Free-Modification Problems are more difficult than their counterparts.

- $\mathcal{F}$ is infinite but contains simple structure like C_4

$\mathcal{F}$-Free-Modification Problems are more difficult than their counterparts.

— $\mathcal{F}$ is infinite but contains simple structure like C_4

CHORDAL CONTRACTION is W[2]-hard.

$\mathcal{F}$-Free-Modification Problems are more difficult than their counterparts.

— $\mathcal{F}$ is infinite but contains simple structure like C_4

CHORDAL CONTRACTION is W[2]-hard.

— $\mathcal{F}$ is finite like $\{ C_4, C_5, 2K_2 \}$

$\mathcal{F}$-Free-Modification Problems are more difficult than their counterparts.

— $\mathcal{F}$ is infinite but contains simple structure like C_4

CHORDAL CONTRACTION is W[2]-hard.

— $\mathcal{F}$ is finite like $\{C_4, C_5, 2K_2\}$

SPLIT CONTRACTION is W[1]-hard.

$\mathcal{F}$-Free-Modification Problems are more difficult than their counterparts.

— $\mathcal{F}$ is infinite but contains simple structure like C_4

CHORDAL CONTRACTION is W[2]-hard.

— $\mathcal{F}$ is finite like $\{ C_4, C_5, 2K_2 \}$

SPLIT CONTRACTION is W[1]-hard.

— $\mathcal{F}$ is simply $\{ P_3 \}$

$\mathcal{F}$-Free-Modification Problems are more difficult than their counterparts.

- $\mathcal{F}$ is infinite but contains simple structure like C_4

  CHORDAL CONTRACTION is W[2]-hard.

- $\mathcal{F}$ is finite like $\{ C_4, C_5, 2K_2 \}$

  SPLIT CONTRACTION is W[1]-hard.

- $\mathcal{F}$ is simply $\{ P_3 \}$

  CLIQUE CONTRACTION is FPT but no poly kernel.

$\mathcal{F}$-Free-Modification Problems are more difficult than their counterparts.

No FPT algorithm for CHORDAL CONTRACTION , SPLIT CONTRACTION

No poly kernel for CLIQUE CONTRACTION

$\mathcal{F}$-Free-Modification Problems are more difficult than their counterparts.

No FPT algorithm for CHORDAL CONTRACTION, SPLIT CONTRACTION

Can we have α-FPT approximation algorithms for these problems?

No poly kernel for CLIQUE CONTRACTION

$\mathcal{F}$-Free-Modification Problems are more difficult than their counterparts.

No FPT algorithm for CHORDAL CONTRACTION, SPLIT CONTRACTION

Can we have α-FPT approximation algorithms for these problems?

No poly kernel for CLIQUE CONTRACTION

Can we have α-lossy kernel for this problem?

$\mathcal{F}$-Free-Modification Problems are more difficult than their counterparts.

No FPT algorithm for CHORDAL CONTRACTION , SPLIT CONTRACTION

Can we have α-FPT approximation algorithms for these problems?

No poly kernel for CLIQUE CONTRACTION

Can we have α-lossy kernel for this problem?

$\alpha > 1$

$\alpha > 1$

α-FPT approximation algorithms

α-lossy kernel

$\alpha > 1$

α-FPT approximation algorithms

* Runs in FPT time
* If there is a solution X of size at most k
  then returns a solution of size at most $\alpha|X|$.

α-lossy kernel

$\alpha > 1$

α-FPT approximation algorithms

* Runs in FPT time

* If there is a solution X of size at most k
  then returns a solution of size at most $\alpha|X|$.

α-lossy kernel

* On input (G, k) produces output (G', k') in polynomial time.

* Given c-factor solution S' for (G', k') produces
  an (α c)-factor solution S for (G, k) in polynomial time.

* Allowed to fail if S' is really bad.

α-FPT approximation algorithms

α-lossy kernel

α-FPT approximation algorithms

$\alpha > 1$ there is **no** α-FPT approx algo for **CHORDAL CONTRACTION** unless $FPT \neq W[1]$.

α-lossy kernel

α-FPT approximation algorithms

$\alpha > 1$ there is **no** α-FPT approx algo for **CHORDAL CONTRACTION** unless FPT $\neq$ W[1].

α-lossy kernel

$\alpha > 1$ there is an α-lossy kernel of poly size for **CLIQUE CONTRACTION**.

α-FPT approximation algorithms

$\alpha > 1$ there is **no** α-FPT approx algo for **CHORDAL CONTRACTION** unless FPT $\neq$ W[1].

α-lossy kernel

$\alpha > 1$ there is an α-lossy kernel of poly size for **CLIQUE CONTRACTION**.

(Poly size — exponent depends on α)

α-FPT approximation algorithms

$\alpha > 1$ there is **no** α-FPT approx algo for CHORDAL CONTRACTION unless FPT $\neq$ W[1].

α-lossy kernel

$\alpha > 1$ there is an α-lossy kernel of poly size for CLIQUE CONTRACTION.

$\alpha > 2$ there is an α-lossy kernel of poly size for SPLIT CONTRACTION.

(Poly size — exponent depends on α)

α-FPT approximation algorithms

$\alpha > 1$ there is **no** α-FPT approx algo for CHORDAL CONTRACTION unless FPT $\neq$ W[1].

$\alpha > 2$ there is an α-FPT approx algo for SPLIT CONTRACTION.

α-lossy kernel

$\alpha > 1$ there is an α-lossy kernel of poly size for CLIQUE CONTRACTION.

$\alpha > 2$ there is an α-lossy kernel of poly size for SPLIT CONTRACTION.

(Poly size — exponent depends on α)

α-FPT approximation algorithms

$\alpha > 1$ there is **no** α-FPT approx algo for CHORDAL CONTRACTION unless FPT $\neq$ W[1].

$\alpha > 2$ there is an α-FPT approx algo for SPLIT CONTRACTION.

$1.25 > \alpha$ there is **no** α-FPT approx algo for SPLIT CONTRACTION under Gap-ETH.

α-lossy kernel

$\alpha > 1$ there is an α-lossy kernel of poly size for CLIQUE CONTRACTION.

$\alpha > 2$ there is an α-lossy kernel of poly size for SPLIT CONTRACTION.

(Poly size — exponent depends on α)

$\alpha > 1$ there is an α-lossy kernel of poly size for CLIQUE CONTRACTION.

Theorem : For any $\alpha > 1$, CLIQUE CONTRACTION parameterised by the size of solution k admits an α-lossy kernel with $O(k^{d+1})$ vertices where $d = 1/\alpha$.

$1.25 > \alpha$ there is no α-FPT approx algo for SPLIT CONTRACTION under Gap-ETH.

Theorem : Assuming Gap-ETH, no FPT algorithm can approximate SPLIT CONTRACTION within a factor of α, for any $\alpha < 1.25$.

CLIQUE CONTRACTION

(α -lossy kernel of polynomial size)

CLIQUE CONTRACTION

Input : Graph G, integer k **Para** : k

Output : Output a set F of edges of minimum cardinality s.t. G/F is a clique.

CLIQUE CONTRACTION

Input : Graph G, integer k

Para : k

Output : Output a set F of edges of minimum cardinality s.t. G/F is a clique.

Every solution of size larger than k is equally bad.

* allowed to fail (see board)
* w.l.o.g G is connected and has at least $(k + 3)$ vertices
* any spanning tree is a trivial solution
* Trivial "no" instance : (Path on four vertices, 1)

Observation : If G can be converted into a clique by **k-edge contraction** then it can also be converted into a clique by **2k-vertex deletion**.

Observation : If G can be converted into a clique by k-edge contraction then it can also be converted into a clique by 2k-vertex deletion.

* Every P_3 is affected by contracted edges

Observation : If G can be converted into a clique by **k-edge contraction** then it can also be converted into a clique by **2k-vertex deletion**.

* Every P_3 is affected by contracted edges
* Every P_3 will be destroyed by deleting end-points of contracted edge.

Observation : If G can be converted into a clique by **k-edge contraction** then it can also be converted into a clique by **2k-vertex deletion**.

- Every P_3 is affected by contracted edges
- Every P_3 will be destroyed by deleting end-points of contracted edge.

Reduction Rule 1 : Given (G, k) find minimum sized X such that $G-X$ is a clique. If $|X| > 4k$ then return no instance.

Observation : If G can be converted into a clique by **k-edge contraction** then it can also be converted into a clique by **2k-vertex deletion**.

* Every P_3 is affected by contracted edges
* Every P_3 will be destroyed by deleting end-points of contracted edge.

Reduction Rule 1 : Given (G, k) find minimum sized X such that G-X is a clique. If $|X| > 4k$ then return no instance.

* 2-factor approx algo for vertex cover in compliment graph

Reduction Rule 1 : Given (G, k) find minimum sized X such that $G-X$ is a clique. If $|X| > 4k$ then return no instance.

Solution lifting algorithm :

- If un-changed, return same solution.
- If no instance then return a spanning tree.

Reduction Rule 1 : Given (G, k) find minimum sized X such that $G-X$ is a clique. If $|X| > 4k$ then return no instance.

Solution lifting algorithm :

- If un-changed, return same solution.
- If no instance then return a spanning tree.

Lemma : RR-1 is 1-safe.

Reduction Rule 1 : Given (G, k) find minimum sized X such that $G-X$ is a clique. If $|X| > 4k$ then return no instance.

Solution lifting algorithm :

- If un-changed, return same solution.
- If no instance then return a spanning tree.

Lemma : RR-1 is 1-safe.

— any solution for reduced instance is large

Reduction Rule 1 : Given (G, k) find minimum sized X such that $G-X$ is a clique. If $|X| > 4k$ then return no instance.

Solution lifting algorithm :

- If un-changed, return same solution.
- If no instance then return a spanning tree.

Lemma : RR-1 is 1-safe.

— any solution for reduced instance is large
— algorithm is allowed to fail

Reduction Rule 1 : Given (G, k) find minimum sized X such that $G-X$ is a clique. If $|X| > 4k$ then return no instance.

Solution lifting algorithm :

- If un-changed, return same solution.
- If no instance then return a spanning tree.

Lemma : RR-1 is 1-safe.

— any solution for reduced instance is large
— algorithm is allowed to fail
— return a large solution (spanning tree)

CLIQUE CONTRACTION :: Partition of V(G) s. t.

1. Each part (called **witness set**) is connected
2. Any two witness sets are adjacent with each other.

CLIQUE CONTRACTION :: Partition of $V(G)$ s. t.

1. Each part (called **witness set**) is connected

2. Any two witness sets are adjacent with each other.

Partition (X, Y) of $V(G)$

- X is small
- Y induces a clique

CLIQUE CONTRACTION :: Partition of V(G) s. t.

1. Each part (called **witness set**) is connected

2. Any two witness sets are adjacent with each other.

Partition (X, Y) of V(G)

- X is small
- Y induces a clique

Role of vertices in Y

CLIQUE CONTRACTION :: Partition of V(G) s. t.

1. Each part (called **witness set**) is connected
2. Any two witness sets are adjacent with each other.

Partition (X, Y) of V(G)

- X is small
- Y induces a clique

Role of vertices in Y

— Provide connectivity to witness sets coming out of X

CLIQUE CONTRACTION :: Partition of V(G) s. t.

1. Each part (called **witness set**) is connected

2. Any two witness sets are adjacent with each other.

Partition (X, Y) of V(G)

- X is small
- Y induces a clique

Role of vertices in Y

— Provide connectivity to witness sets coming out of X

— Dictate witness sets in X

Role of vertices in Y

- Provide connectivity to witness sets coming out of X

- Dictate witness sets in X

Role of vertices in Y

— Provide connectivity to witness sets coming out of X

For every subset X' of X, mark a vertex which is common neighbour.

— Dictate witness sets in X

Role of vertices in Y

— Provide connectivity to witness sets coming out of X

For every subset X′ of X, mark a vertex which is common neighbour.

— Dictate witness sets in X

For every subset X′ of X, mark (2k + 1) vertices which are common non-neighbour.

Role of vertices in Y

— Provide connectivity to witness sets coming out of X

For every subset X' of X, mark a vertex which is common neighbour.

— Dictate witness sets in X

For every subset X' of X, mark $(2k + 1)$ vertices which are common non-neighbour.

Kernel of size $O(2^k)$ for CLIQUE CONTRACTION.

Role of vertices in Y

$$\frac{d+1}{d} \leq \alpha$$

- Provide connectivity to witness sets coming out of X

  For every subset X' of X, mark a vertex which is common neighbour.

  For every subset X' of X **of size <= d**, mark a vertex which is common neighbour.

- Dictate witness sets in X

  For every subset X' of X, mark (2k + 1) vertices which are common non-neighbour.

  For every subset X' of X **of size <= d**, mark (2k + 1) vertices which are common non-neighbour.

For every subset X' of X of size <= d, mark a vertex which is common neighbour.

$$\frac{d+1}{d} \le \alpha$$

For every subset X' of X of size <= d, mark (2k + 1) vertices which are common non-neighbour.

Reduction Rule 2 : Mark vertices in Y and delete unmarked vertices.

RR-2 runs in poly-time on large instances.

Reduction Rule 2 : Mark vertices in Y and delete unmarked vertices.

Y' — deleted vertices. $(G' = G \setminus Y', k)$ is new instance.

F' is given solution for (G', k)

Solution Lifting Algorithm: Add a vertex in Y to witness sets completely in X which are of size $\geq d$.

Lemma : Size of obtained solution F is $\leq \alpha |F'|$.

Y' — deleted vertices. $(G' = G \setminus Y', k)$ is new instance.

F' is given solution for (G', k)

Solution Lifting Algorithm: Add a vertex in Y to witness sets completely in X which are of size >= d.

Lemma : Size of obtained solution F is $\le \alpha |F'|$.

— Such vertex exists because of first marking

— Adding an extra edge for every d vertices

— Every witness set of size <=d in X is adjacent with everything in Y' because of second marking

Lemma-1 : Size of obtained solution F is $\leq \alpha |F'|$.

$$(G, k) \longrightarrow (G', k)$$

$\boxed{\textbf{Lemma-2} : \text{Optimum does not increase. i.e. } OPT(G', k) \leq OPT(G, k).}$

Lemma-1 : Size of obtained solution F is $\leq \alpha |F'|$.

$$(G, k) \longrightarrow (G', k)$$

Lemma-2 : Optimum does not increase. i.e. $OPT(G', k) \leq OPT(G, k)$.

— Witness sets completely in $Y \setminus Y'$

Lemma-1 : Size of obtained solution F is $<= \alpha |F'|$.

$$(G, k) \longrightarrow (G', k)$$

$\boxed{\text{Lemma-2 : Optimum does not increase. i.e. } OPT(G', k) <= OPT(G, k).}$

— Witness sets completely in $Y \setminus Y'$

— Witness sets intersecting Y and $Y \setminus Y'$

Lemma-1 : Size of obtained solution F is $\leq \alpha |F'|$.

$$(G, k) \longrightarrow (G', k)$$

$\boxed{\textbf{Lemma-2} : \text{Optimum does not increase. i.e. } OPT(G', k) \leq OPT(G, k).}$

- Witness sets completely in $Y \setminus Y'$
- Witness sets intersecting Y and $Y \setminus Y'$
- Witness sets intersecting X and $Y \setminus Y'$

Reduction Rule 2 : Given (G, k), mark vertices in Y and delete unmarked vertices.

Solution Lifting Algorithm: Add a vertex in Y to witness sets completely in X which are of size $>= d$.

Lemma-1 : Size of obtained solution F is $<= \alpha\, |F'|$.

Lemma-2 : Optimum does not increase. i.e. $OPT(G', k) <= OPT(G, k)$.

Lemma : RR-2 is α safe.

Reduction Rule 2 : Given (G, k), mark vertices in Y and delete unmarked vertices.

Solution Lifting Algorithm: Add a vertex in Y to witness sets completely in X which are of size >= d.

Lemma-1 : Size of obtained solution F is $\leq \alpha |F'|$.

Lemma-2 : Optimum does not increase. i.e. $OPT(G', k) \leq OPT(G, k)$.

Lemma : RR-2 is α safe.

$$\frac{|F|}{OPT(G, k)} \leq \alpha \frac{|F'|}{OPT(G', k)}$$

Theorem : For any $\alpha > 1$, CLIQUE CONTRACTION parameterised by the size of solution k admits an α-lossy kernel with $O(k^{d+1})$ vertices where $d = 1/\alpha$.

Theorem : For any $\alpha > 2$, SPLIT CONTRACTION parameterised by the size of solution k admits an α-lossy kernel with $O(k^{poly(d)})$ vertices where $d = 1/\alpha$.

SPLIT CONTRACTION

(Connecting α -FPT approx algo with α-lossy kernel)

Assume α-lossy kernel of size $f(k)$

$$(G, k) \longrightarrow (G', k')$$

$$F \longleftarrow F'$$

s.t. if F' is c-factor solution then F is $(\alpha\, c)$-factor solution.

Assume α-lossy kernel of size $f(k)$

$$(G, k) \longrightarrow (G', k')$$

$$F \longleftarrow F'$$

s.t. if F' is c-factor solution then F is $(\alpha\, c)$-factor solution.

Compute optimum solution for (G', k') in FPT time.

Assume α-lossy kernel of size $f(k)$

$$(G, k) \longrightarrow (G', k')$$

$$F \longleftarrow F'$$

s.t. if F' is c-factor solution then F is $(\alpha\ c)$-factor solution.

Compute optimum solution for (G', k') in FPT time.

Compute α-factor solution for (G, k) in FPT time.

Assume α-lossy kernel of size f(k)

$$(G, k) \longrightarrow (G', k')$$

$$F \longleftarrow F'$$

s.t. if F′ is c-factor solution then F is (α c)-factor solution.

Compute optimum solution for (G′, k′) in FPT time.

Compute α-factor solution for (G, k) in FPT time.

α-factor FPT approximation algorithm for SPLIT CONTRACTION.

Assume α-lossy kernel of size f(k)

α-factor FPT approximation algorithm for SPLIT CONTRACTION.

If there is a problem which

— does not admit an α-factor FPT approximation algorithm

— has a gap preserving reduction to SPLIT CONTRACTION

then no α-lossy kernel of any size exists.

DENSEST k SUBGRAPH

Input : Graph G, integers k, t and constants $\varepsilon < 1 < \beta$

Parameters : k + t

Output : Are there at least β k vertices which spans ε t many edges?

Guarantee : There is a cliques of size k in G.

DENSEST k SUBGRAPH

Input : Graph G, integers k, t and constants $\varepsilon < 1 < \beta$

Parameters : $k + t$

Output : Are there at least βk vertices which spans εt many edges?

Guarantee : There is a cliques of size k in G.

Theorem (Chalermsook et al. FOCS 2017) : Assuming Gap-ETH, DENSEST k SUBGRAPH can not be solved in time $f(k, t)\mathrm{poly}(n)$ for any function f.

Theorem (Chalermsook et al. FOCS 2017) : Assuming Gap-ETH, DENSEST k SUBGRAPH can not be solved in time $f(k, t)\text{poly}(n)$ for any function f.

DENSEST k SUBGRAPH $\longrightarrow$ SPLIT CONTRACTION

$(G, k, t, \varepsilon, \beta)$ $\quad\quad$ (G', k')

Theorem (Chalermsook et al. FOCS 2017) : Assuming Gap-ETH, DENSEST k SUBGRAPH can not be solved in time f(k, t)poly(n) for any function f.

DENSEST k SUBGRAPH $\longrightarrow$ SPLIT CONTRACTION

$(G, k, t, \varepsilon, \beta)$ $\qquad$ (G', k')

— Runs in FPT time.

Theorem (Chalermsook et al. FOCS 2017) : Assuming Gap-ETH, DENSEST k SUBGRAPH can not be solved in time $f(k, t)\mathrm{poly}(n)$ for any function f.

DENSEST k SUBGRAPH → SPLIT CONTRACTION

$(G, k, t, \varepsilon, \beta)$ → (G', k')

— Runs in FPT time.

— G' can be contracted to a split graph with k' edge contractions (because of guarantee)

Theorem (Chalermsook et al. FOCS 2017) : Assuming Gap-ETH, DENSEST k SUBGRAPH can not be solved in time $f(k, t)\mathrm{poly}(n)$ for any function f.

DENSEST k SUBGRAPH $\longrightarrow$ SPLIT CONTRACTION

$(G, k, t, \varepsilon, \beta)$ $\qquad$ (G', k')

— Runs in FPT time.

— G' can be contracted to a split graph with k' edge contractions (because of guarantee)

— For $1.25 > \alpha$, given an α-factor solution for SPLIT CONTRACTION, we can obtain a solution for DENSEST k SUBGRAPH in poly-time.

Theorem : Assuming Gap-ETH, no FPT algorithm can approximate SPLIT CONTRACTION within a factor of α, for any $\alpha < 1.25$.

To sum up...

* Allowing only "edge contraction" makes Graph Modification Problems harder.

  — FPT algorithm
  — (classical) kernels
  — Exact algorithm.

* Can we obtain FPT approx algorithm and lossy kernels?

Other successes

* Lossy kernels for

  — TREE CONTRACTION (KMRT, FSTTCS 16)
  — BOUNDED TREE CONTRACTION (ALST, CIAC 17)
  — (Tree + q edges) CONTRACTION (AST, IPEC 17)
  — CACTUS CONTRACTION (KRT, COCOON 18)

* Exact Algorithms

  — PATH CONTRACTION (ADFST, ICALP 19)

Open Problems regarding Contraction

* (No) kernels + lossy kernels

  — Planar graphs
  — Bi-partite graphs

  — Outer-planar graphs
  — Graphs of tw <= 2

* FPT algorithm for

  — Interval graphs

  — Outer-planar graphs
  — Graph of tw <= 2

* Exact algorithms for

  — Tree Contraction
  — Clique Contraction *

Thank you.

(Any Questions?)