# Lossy Kernels for Graph Contraction Problems

### Prafullkumar Tale The Institute Of Mathematical Sciences, Chennai, India

# Graph Modification Problems

#### **H-Modification Problems**

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

#### Para : k

# Graph Modification Problems

#### **H-Modification Problems**

### Input : Graph G, integer k Output: Can we make at most k modifications in G so that resulting graph is in H?

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

#### Para:k

# Graph Modification Problems

### **H-Modification Problems**

### Input : Graph G, integer k Output : Can we make at most k modifications in G so that resulting graph is in H?

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

#### Modification -(1) vertex deletion (2) edge deletion (3) edge addition

#### Para:k

# (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 = $\mathcal{F}$ -Free-Modification Problems			
Modification — (1) vertex deletion (3) edge addition		(2) edge deletion (4) edge contraction	
Problem	Ħ	F-Free	Modification
VERTEX COVER	Empty Graphs	P2	Vertex Deletion
EDGE BIPARTIZATION	Bipartite Graphs	C3, C5, C6,	Edge Deletion
MINIMUM FILL-IN	Chordal Graphs	C4, C5, C6,	Edge Addition
CLUSTER EDITING	Set of Cliques	P3	Edge Add + Pel.



# 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.

# (2) edge deletion(4) edge contraction

# If F is infinite and (1), (2), and/or (3) then F-Free - Modification Problems are FPT. • MINIMUM FILL-IN is FPT (Cai '96 + Kaplan '96).

EDGE BIPARTIZATION is FPT (S. Wernicke 2003).

# (2) edge deletion(4) edge contraction

### 6 + Kaplan '96). Vernicke 2003).

#### F-Free - Modification Problems when only (4) is allowed?

## (2) edge deletion(4) edge contraction

#### F-Free - Modification Problems when only (4) is allowed?

**4**k

#### CLIQUE CONTRACTION exp(klog(k))

#### PLANAR CONTRACTION

#### BIPARTITE CONTRACTION

FPT

 $exp(k^2)$ 

#### (2) edge deletion (4) edge contraction

No poly kernel

Heggernes et al. (2011)

No poly kernel

Cai et al. (2013)

Golavach et al. (2013)

Guillemot + Mark (2013)



F-Free - Modification Problems when only (4) is allowed?

Can we have generic theorem as that of Cai 1996?

#### (2) edge deletion (4) edge contraction

F-Free - Modification Problems when only (4) is allowed?

Can we have generic theorem as that of Cai 1996?

#### (2) edge deletion (4) edge contraction

#### NO

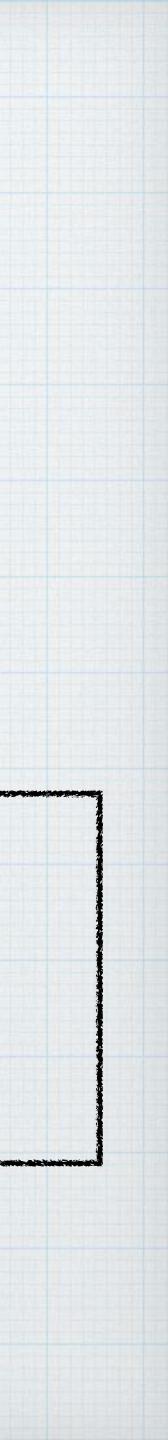
F-Free - Modification Problems when only (4) is allowed?

Can we have generic theorem as that of Cai 1996?

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

#### (2) edge deletion (4) edge contraction

### 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.

#### 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 WE11-hard.



#### — F is infinite but contains simple structure like C4

#### — F is infinite but contains simple structure like C4

### CHORDAL CONTRACTION is W[2]-hard.

CHORDAL CONTRACTION is W[2]-hard.

### $-\mathcal{F}$ is finite like { C4, C5, 2K2 }

CHORDAL CONTRACTION is W[2]-hard.

 $-\mathcal{F}$  is finite like { C4, C5, 2K2 }

SPLIT CONTRACTION is W[1]-hard.

CHORDAL CONTRACTION is W[2]-hard.

 $-\mathcal{F}$  is finite like { C4, C5, 2K2 }

SPLIT CONTRACTION is W[1]-hard.

- F is simply {P3}

CHORDAL CONTRACTION is W[2]-hard.

 $-\mathcal{F}$  is finite like { C4, C5, 2K2 }

SPLIT CONTRACTION is WE11-hard.

- F is simply {P3}

CLIQUE CONTRACTION is FPT but no poly kernel.

### No FPT algorithm for CHORDAL CONTRACTION, SPLIT CONTRACTION

### No poly kernel for CLIQUE CONTRACTION



### No FPT algorithm for CHORDAL CONTRACTION, SPLIT CONTRACTION

### Can we have $\alpha$ -FPT approximation algorithms for these problems?

### No poly kernel for CLIQUE CONTRACTION

### No FPT algorithm for CHORDAL CONTRACTION, SPLIT CONTRACTION

### Can we have $\alpha$ -FPT approximation algorithms for these problems?

### No poly kernel for CLIQUE CONTRACTION

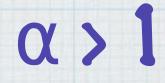
### Can we have $\alpha$ -lossy kernel for this problem?

### No FPT algorithm for CHORDAL CONTRACTION, SPLIT CONTRACTION

### Can we have $\alpha$ -FPT approximation algorithms for these problems?

### No poly kernel for CLIQUE CONTRACTION

### Can we have $\alpha$ -lossy kernel for this problem?



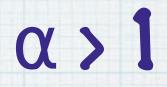
#### α-lossy kernel



### $\alpha$ -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  $\alpha$ IXI.

α-lossy kernel



### $\alpha$ -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  $\alpha$ IXI.
- α-lossy kernel
- \* On input (G, k) produces output (G', k') in polynomial time.
- \* Given c-factor solution S' for (G', K') produces an  $(\alpha c)$ -factor solution S for (G, k) in polynomial time.
- \* Allowed to fail if S' is really bad.



#### α-lossy kernel

#### $\alpha$ > 1 there is no $\alpha$ -FPT approx algo for CHORPAL CONTRACTION unless FPT $\neq$ WE11.

#### α-lossy kernel



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

#### α-lossy kernel

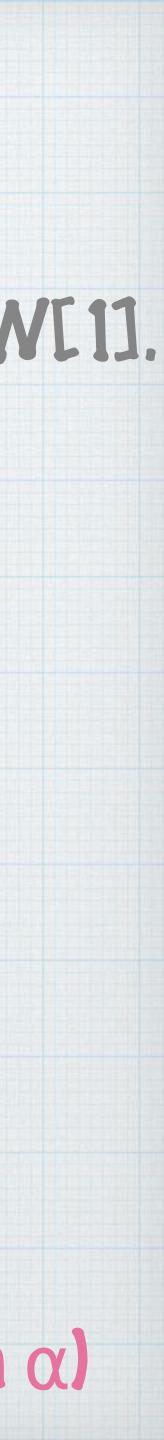
#### $\alpha$ > 1 there is an $\alpha$ -lossy kernel of poly size for CLIQUE CONTRACTION.



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

#### α-lossy kernel

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



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

#### a-lossy kernel

 $\alpha$  > 1 there is an  $\alpha$ -lossy kernel of poly size for CLIQUE CONTRACTION.  $\alpha$  > 2 there is an  $\alpha$ -lossy kernel of poly size for SPLIT CONTRACTION.



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

a-lossy kernel

 $\alpha$  > 1 there is an  $\alpha$ -lossy kernel of poly size for CLIQUE CONTRACTION.  $\alpha$  > 2 there is an  $\alpha$ -lossy kernel of poly size for SPLIT CONTRACTION.

#### $\alpha$ > 1 there is no $\alpha$ -FPT approx algo for CHORDAL CONTRACTION unless FPT $\neq$ WC11.



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

a-lossy kernel

 $\alpha$  > 1 there is an  $\alpha$ -lossy kernel of poly size for CLIQUE CONTRACTION.  $\alpha$  > 2 there is an  $\alpha$ -lossy kernel of poly size for SPLIT CONTRACTION.

### $\alpha$ > 1 there is no $\alpha$ -FPT approx algo for CHORDAL CONTRACTION unless FPT $\neq$ WE11.

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

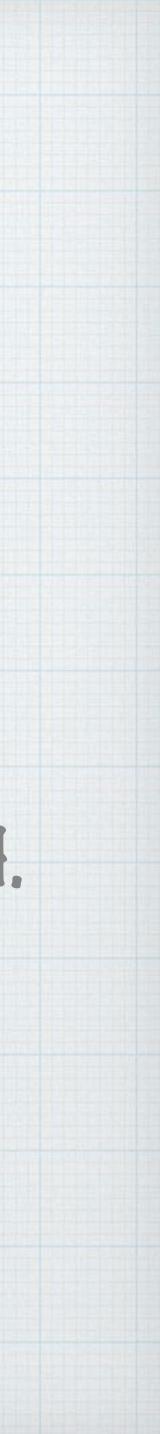


#### $\alpha > 1$ there is an $\alpha$ -lossy kernel of poly size for CLIQUE CONTRACTION.

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

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

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





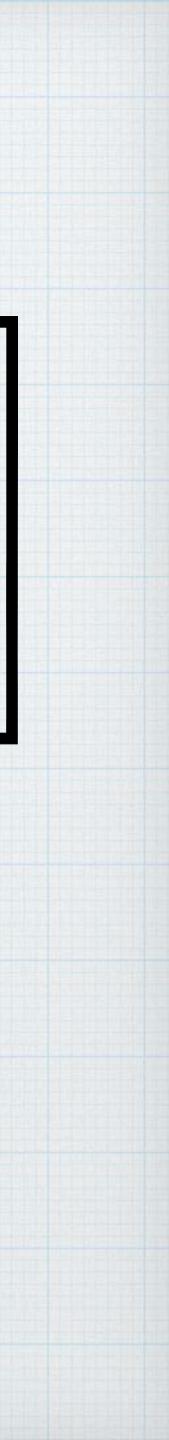
#### ( $\alpha$ -lossy kernel of polynomial size)

#### **CLIQUE CONTRACTION**

### CLIQUE CONTRACTION

### Input : Graph G, integer k Output : Output a set F of edges of minimum cardinality s.t. G/F is a clique.

#### Para:k

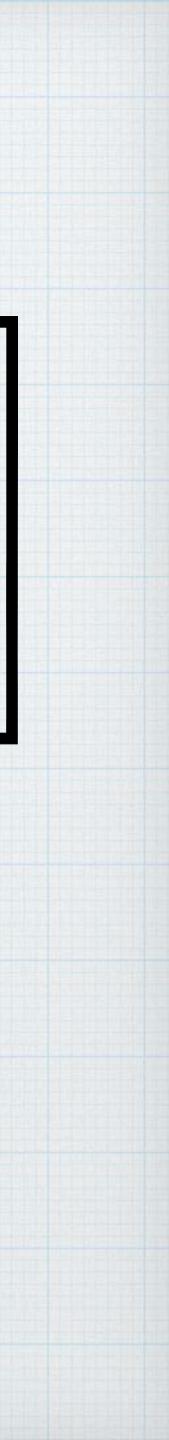


### CLIQUE CONTRACTION

### Input : Graph G, integer 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.

#### Para:k



#### \* 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 P3 is affected by contracted edges

it can also be converted into a clique by 2k-vertex deletion.

#### \* Every P3 is affected by contracted edges \* Every P3 will be destroyed by deleting end-points of contracted edge.

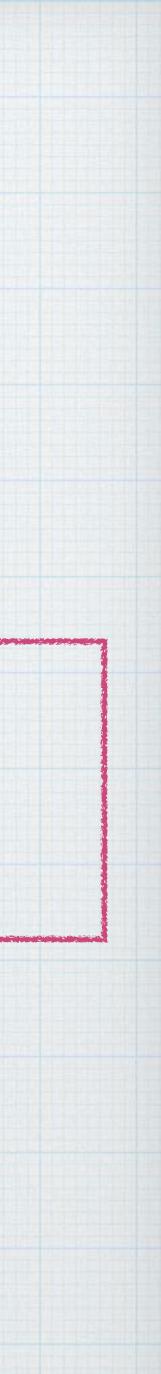
## **Observation :** If G can be converted into a clique by k-edge contraction then

it can also be converted into a clique by 2k-vertex deletion.

\* Every P3 is affected by contracted edges \* Every P3 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 IXI > 4k then return no instance.

## **Observation :** If G can be converted into a clique by k-edge contraction then



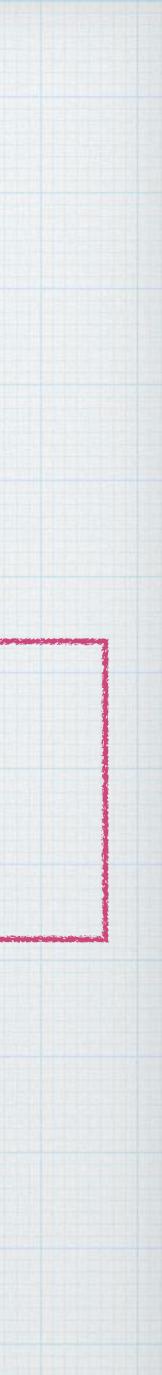
it can also be converted into a clique by 2k-vertex deletion.

\* Every P3 is affected by contracted edges \* Every P3 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 IXI > 4k then return no instance.

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

## **Observation :** If G can be converted into a clique by k-edge contraction then



# Solution lifting algorithm : If un-changed, return same solution. If no instance then return a spanning tree.



Solution lifting algorithm: • If un-changed, return same solution. • If no instance then return a spanning tree.

Lemma : RR-1 is 1-safe.



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



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



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.

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

#### 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.

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

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

• Xissmall • Y induces a clique

### Role of vertices in Y

## CLIQUE CONTRACTION :: Partition of V(G) s. t. Each part (called witness set) is connected 1. 2. Any two witness sets are adjacent with each other. Partition (X, Y) of V(G) • Xissmall • 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. Each part (called witness set) is connected 1. 2. Any two witness sets are adjacent with each other. Partition (X, Y) of V(G) • Xissmall • Y induces a clique Role of vertices in Y - Provide connectivity to witness sets coming out of X - Dictate witness sets in X

#### - Provide connectivity to witness sets coming out of X

#### - Dictate witness sets in X

- 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

- 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

common non-neighbour.

### For every subset X' of X, mark (2k + 1) vertices which are

- Provide connectivity to witness sets coming out of X For every subset X' of X, mark a vertex which is

common neighbour.

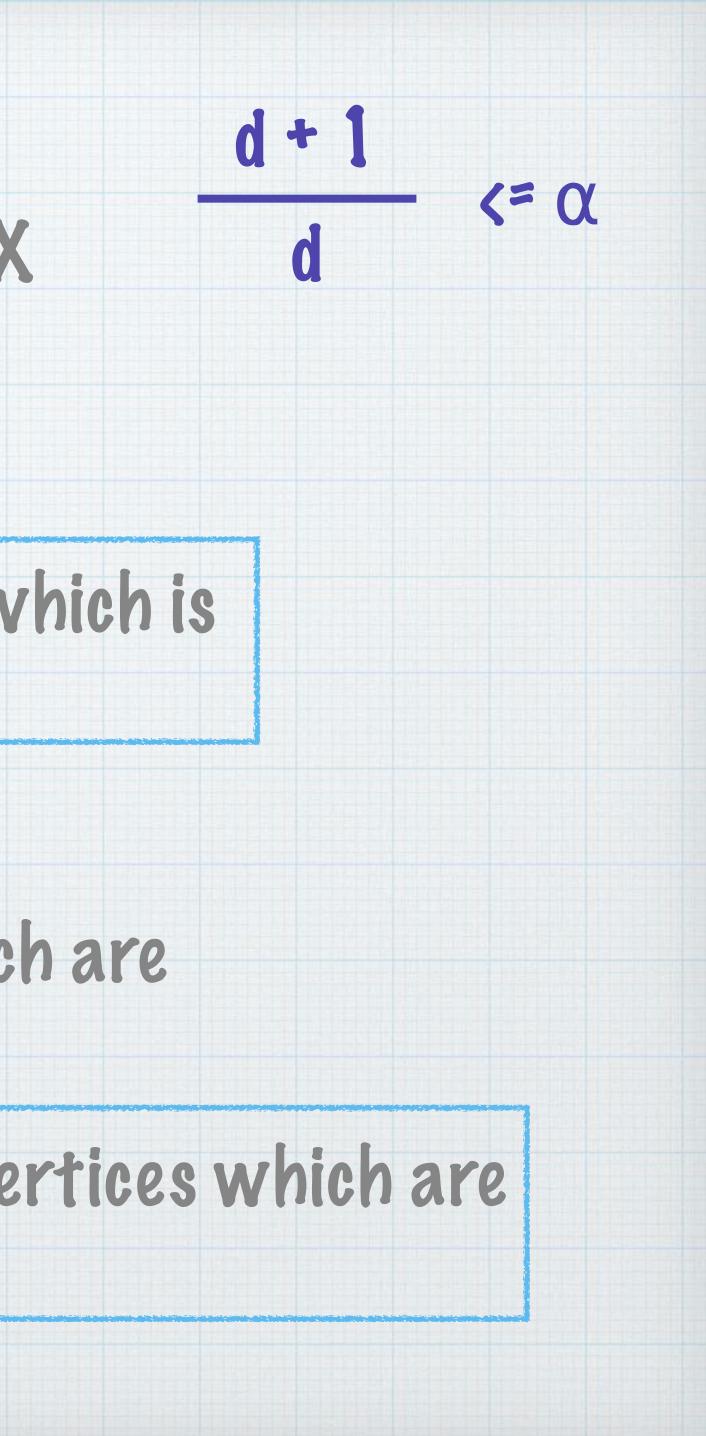
- Dictate witness sets in X

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

Kernel of size O(2<sup>k</sup>) for CLIQUE CONTRACTION.

- 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.

common non-neighbour.



### For every subset X' of X of size <= d, mark (2k + 1) vertices which are

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

#### 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\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 $\zeta = \alpha$ IF'l.



### 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  $\zeta = \alpha$  IF1.

- Such vertex exists because of first marking

— Adding an extra edge for every d vertices

because of second marking

### — Every witness set of size <=d in X is adjacent with everything in Y'</p>



#### Lemma-1 : Size of obtained solution F is $\zeta = \alpha$ IF1.

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

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

#### Lemma-1 : Size of obtained solution F is $\zeta = \alpha$ IF1.

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

### - Witness sets completely in Y\Y'

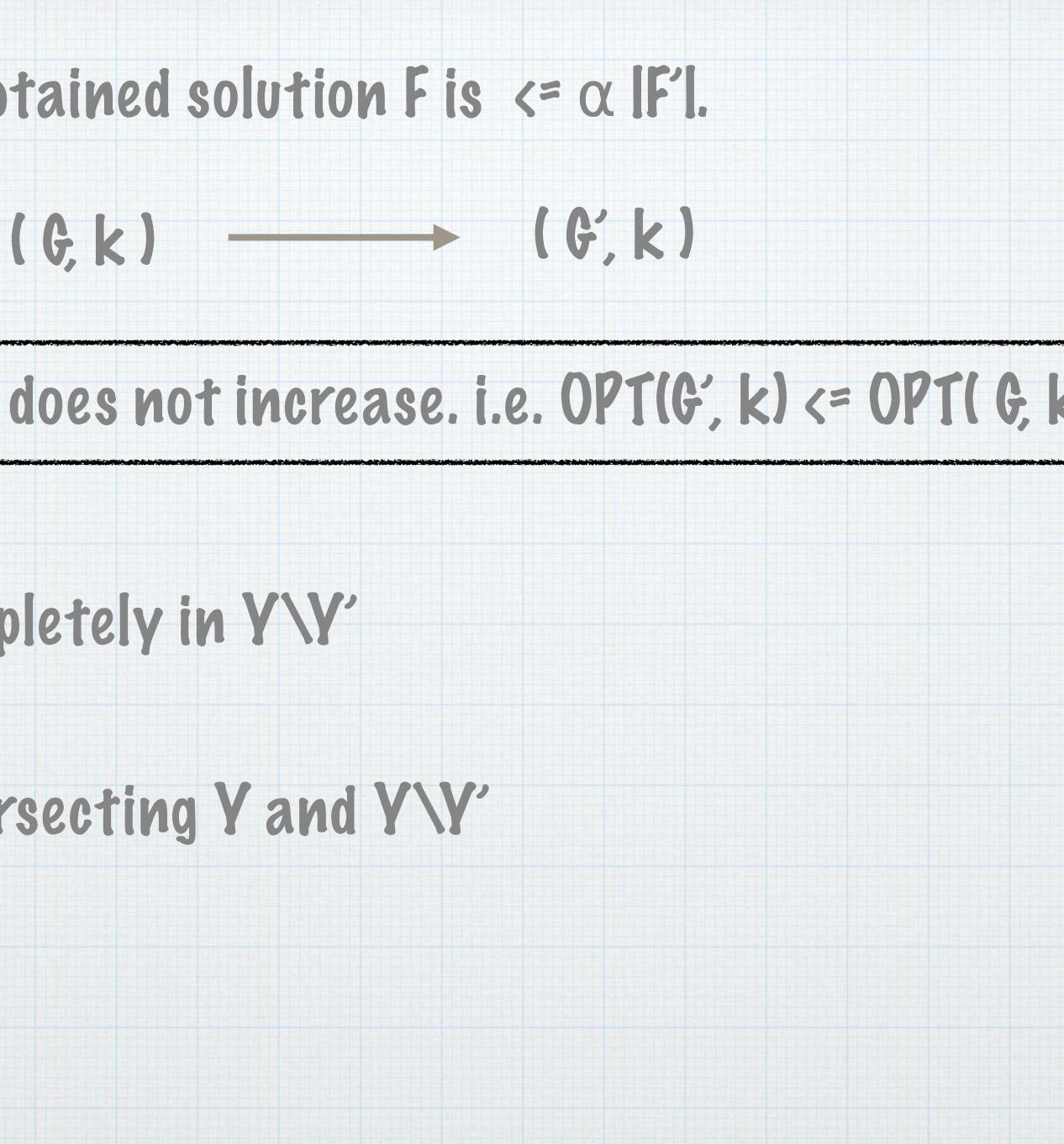
 $(G,k) \rightarrow (G',k)$ 

#### Lemma-1 : Size of obtained solution F is $\zeta = \alpha$ IF1.

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

### - Witness sets completely in Y\Y'

### — Witness sets intersecting Y and Y Y



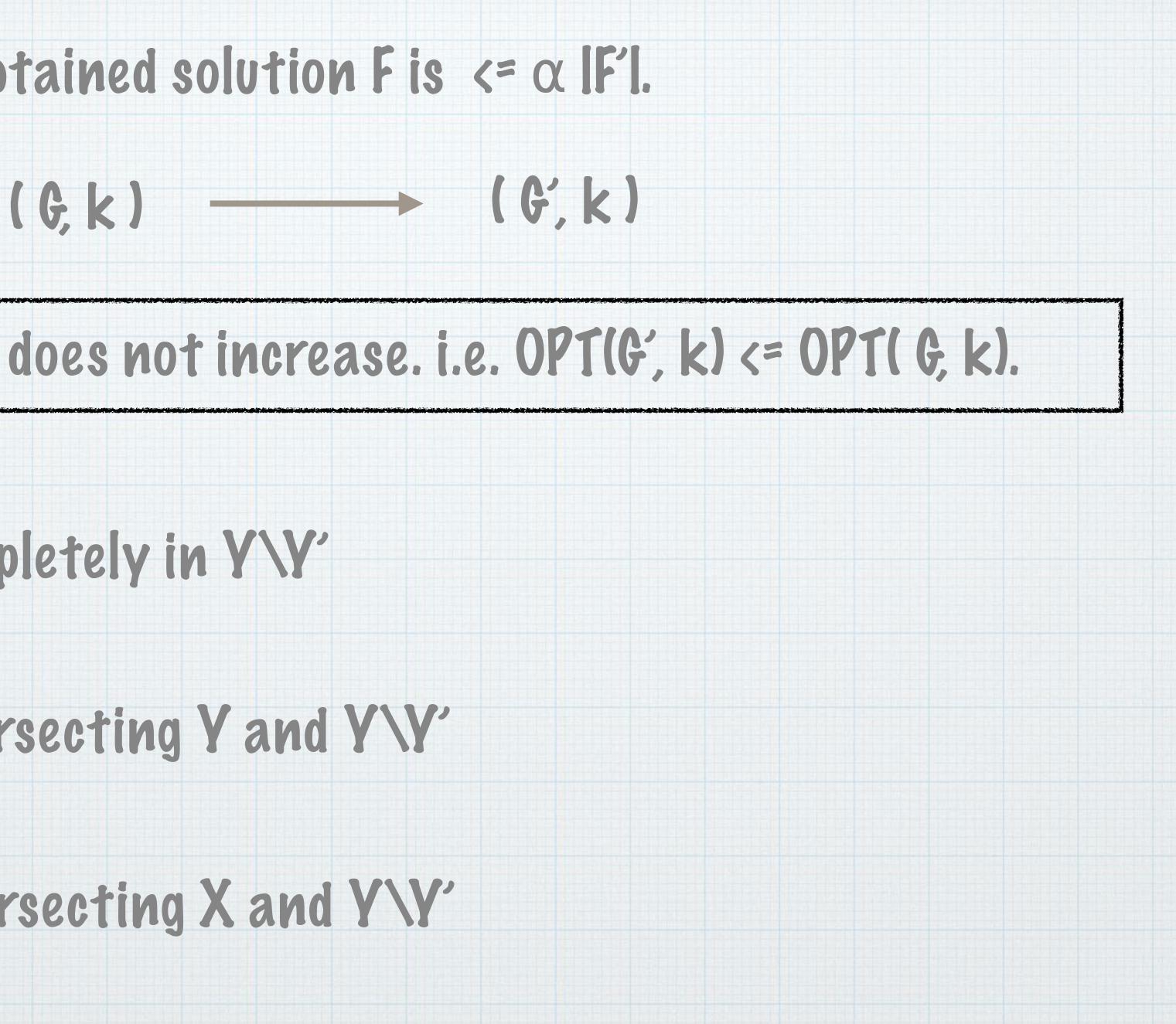
#### Lemma-1 : Size of obtained solution F is $\zeta = \alpha$ [F].

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

### - Witness sets completely in Y\Y'

### - Witness sets intersecting Y and Y\Y'

### - Witness sets intersecting X and Y Y'

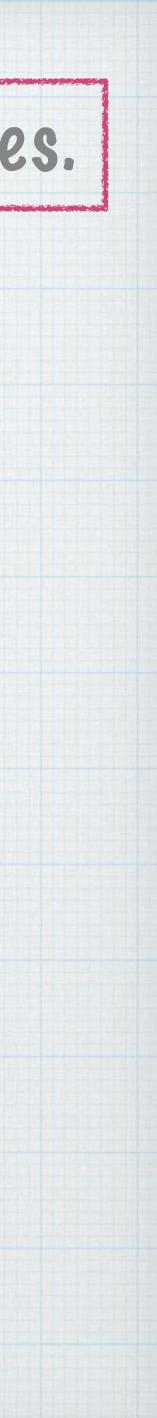


### Reduction Rule 2 : Given (G, k), mark vertices in Y and delete unmarked vertices.

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  $\zeta = \alpha$  [F]. Lemma-2: Optimum does not increase. i.e. OPT(G', k) <= OPT(G, k).

Lemma : RR-2 is  $\alpha$  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  $\zeta = \alpha$  [F]. Lemma-2: Optimum does not increase. i.e. OPT(G', k) <= OPT(G, k).

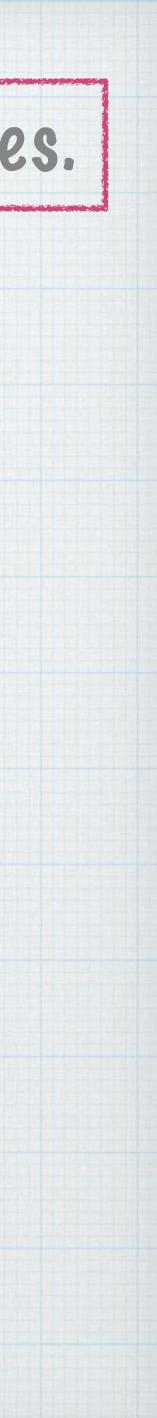
IF

OPT(G,k)

Lemma : RR-2 is  $\alpha$  safe.

OPT(G',k)

IF'I



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

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

#### SPLIT CONTRACTION

#### (Connecting $\alpha$ -FPT approx algo with $\alpha$ -lossy kernel)

# Assume $\alpha$ -lossy kernel of size f(k) $(G,k) \longrightarrow (G',k')$ F F s.t. if F' is c-factor solution then F is ( $\alpha$ c)-factor solution.

F

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

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

F

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

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

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

F

Compute  $\alpha$ -factor solution for (G, k) in FPT time.

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

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.

F

Compute  $\alpha$ -factor solution for (G, k) in FPT time.

 $\alpha$ -factor FPT approximation algorithm for SPLIT CONTRACTION.

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

If there is a problem which

- does not admit an  $\alpha$ -factor FPT approximation algorithm

- has a gap preserving reduction to SPLIT CONTRACTION

then no  $\alpha$ -lossy kernel of any size exists.

#### $\alpha$ -factor FPT approximation algorithm for SPLIT CONTRACTION.

#### **DENSEST K SUBGRAPH**

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

Parameters : k + t

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

#### **Output :** Are there at least $\beta$ k vertices which spans $\varepsilon$ t many edges?



#### **DENSEST K SUBGRAPH**

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

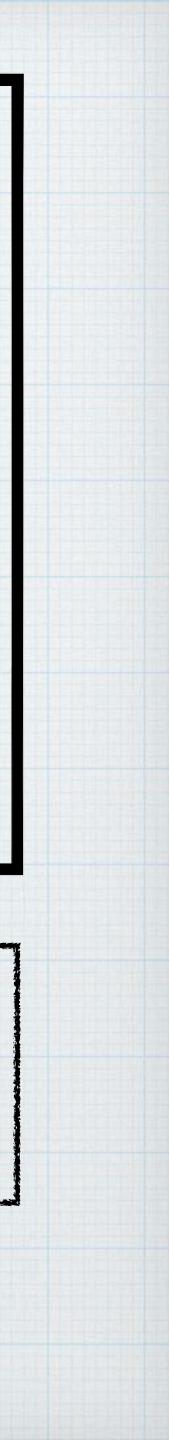
Parameters : k + t

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

SUBGRAPH can not be solved in time f(k, t)poly(n) for any function f.

#### Output : Are there at least $\beta$ k vertices which spans $\varepsilon$ t many edges?

# Theorem (Chalermsook et al. FOCS 2017): Assuming Gap-ETH, PENSEST 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



### SPLIT CONTRACTION





#### 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

(G, k, t, E, B)

#### -Runs in FPT time.

### SPLIT CONTRACTION





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

## **DENSEST k SUBGRAPH** (G, k, t, E, B)

#### - Runs in FPT time.

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

### SPLIT CONTRACTION

(G', K')



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

## **DENSEST k SUBGRAPH** $(G, k, t, \varepsilon, \beta)$

- -Runs in FPT time.
- -G' can be contracted to a split graph with k' edge contractions (because of guarantee)

### SPLIT CONTRACTION

## (G', K')

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



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



## IO SUM UD...

## \* 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 BOUNDED TREE CONTRACTION (KMRT, FSTTCS 16) (ALST, CIAC 17) (AST, IPEC 17) (AST, IPEC 17) (KRT, COCOON 18)

## \* Exact Algorithms

## - PATH CONTRACTION

## (APFST, ICALP 19)

# **Open Problems regarding Contraction**

# \* (No) kernels + lossy kernels

— Planar graphs — Bi-partite graphs

— Outer-planar graphs — Graphs of tw <= 2</p>

## \* FPT algorithm for

- Interval graphs
- Outer-planar graphs — Graph of tw <= 2</p>
- \* Exact algorithms for
  - Tree Contraction — Clique Contraction \*





#### (Any Questions?)