Lossy Kernelization for Some Graph Contraction Problems

R Krithika P. Mishra A. Rai P. Tale

October 25, 2017

The Institute of Mathematical Sciences, HBNI, Chennai, India

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

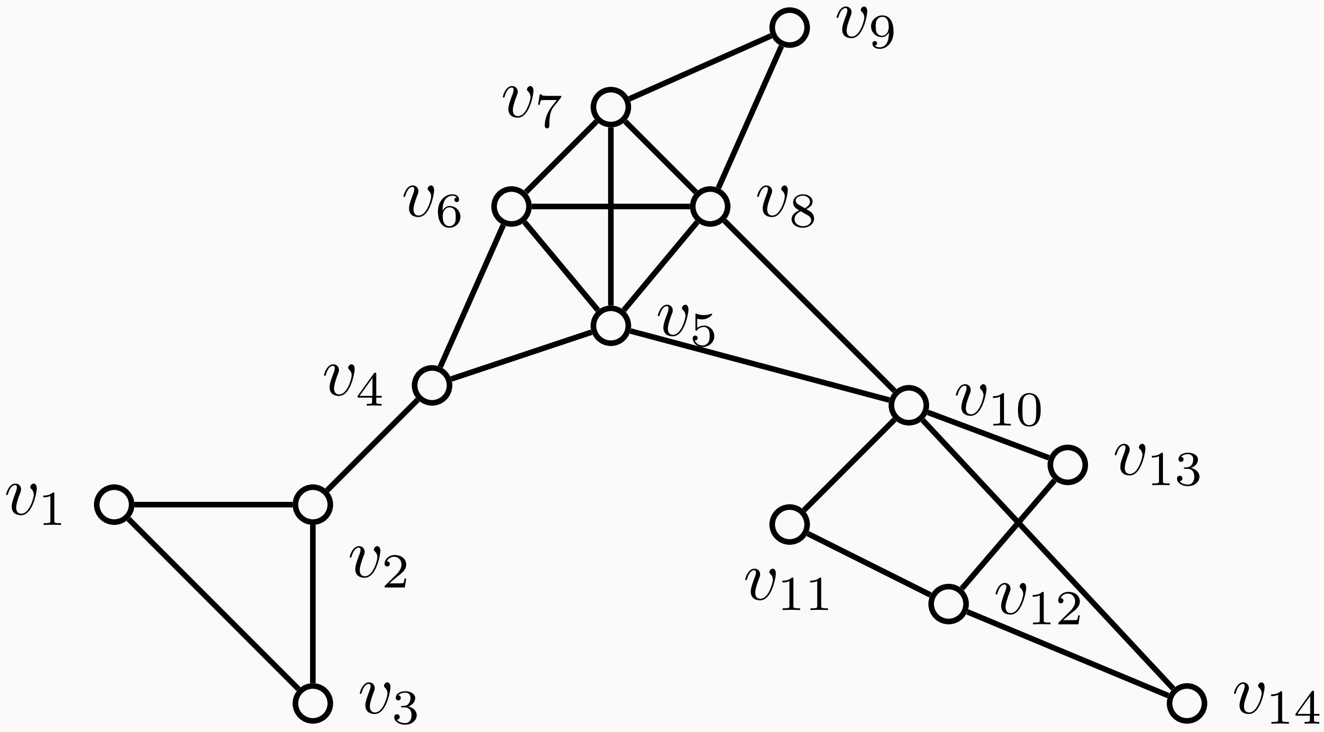
[HvtHL+12]	TREE CONTRACTION PATH CONTRACTION	4^k $2^{k+o(k)}$
[GvtHP13]	PLANAR CONTRACTION	FPT
[CG13]	CLIQUE CONTRACTION	$2^{\mathcal{O}(k \log k)}$
[HvtHLP13] [GM13]	BIPARTITE CONTRACTION	FPT $2^{\mathcal{O}(k^2)}$
[LMS13] [CG13]	$P_{\ell+1}$-FREE CONTRACTION C_ℓ-FREE CONTRACTION	$W[2]$-hard $W[2]$-hard
[ALSZ17]	SPLIT CONTRACTION	$W[2]$-hard

$\mathcal{F}$-Contraction: Kernelization

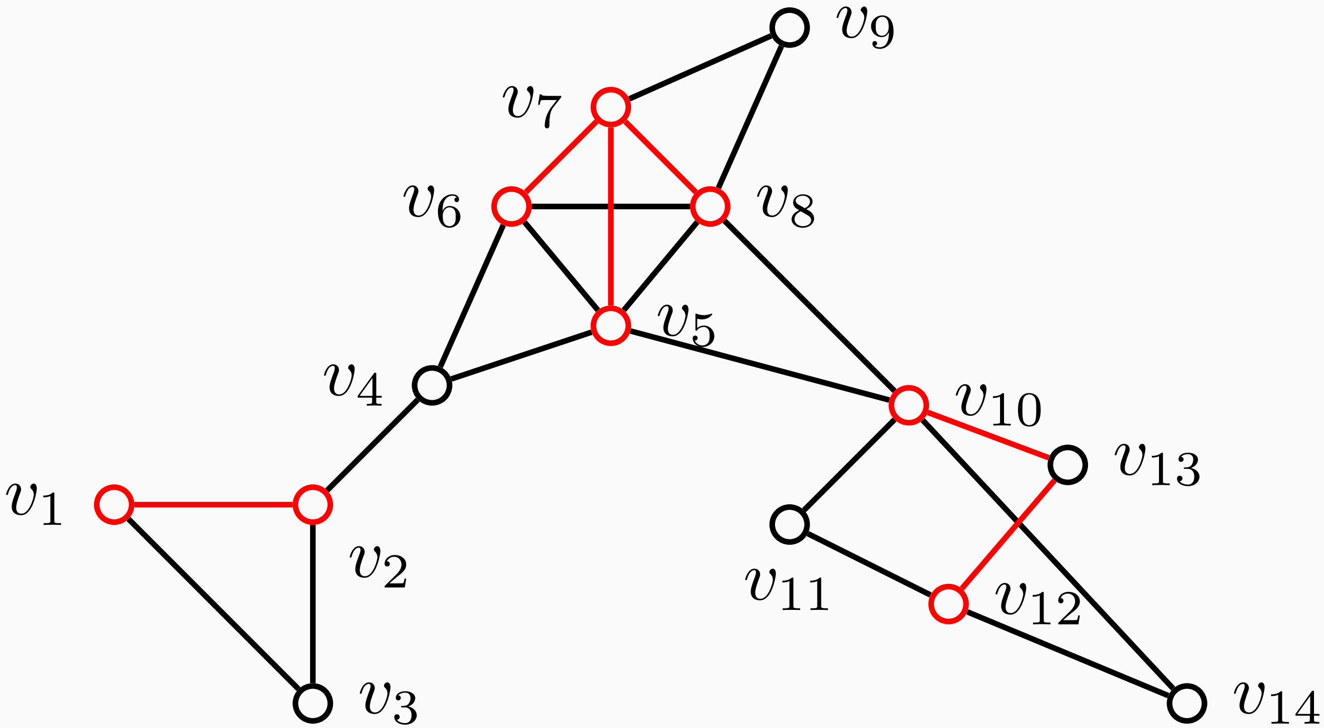
[HvtHL⁺12]	TREE CONTRACTION	No poly-kernel
	PATH CONTRACTION	$\mathcal{O}(k)$

Tree Contraction

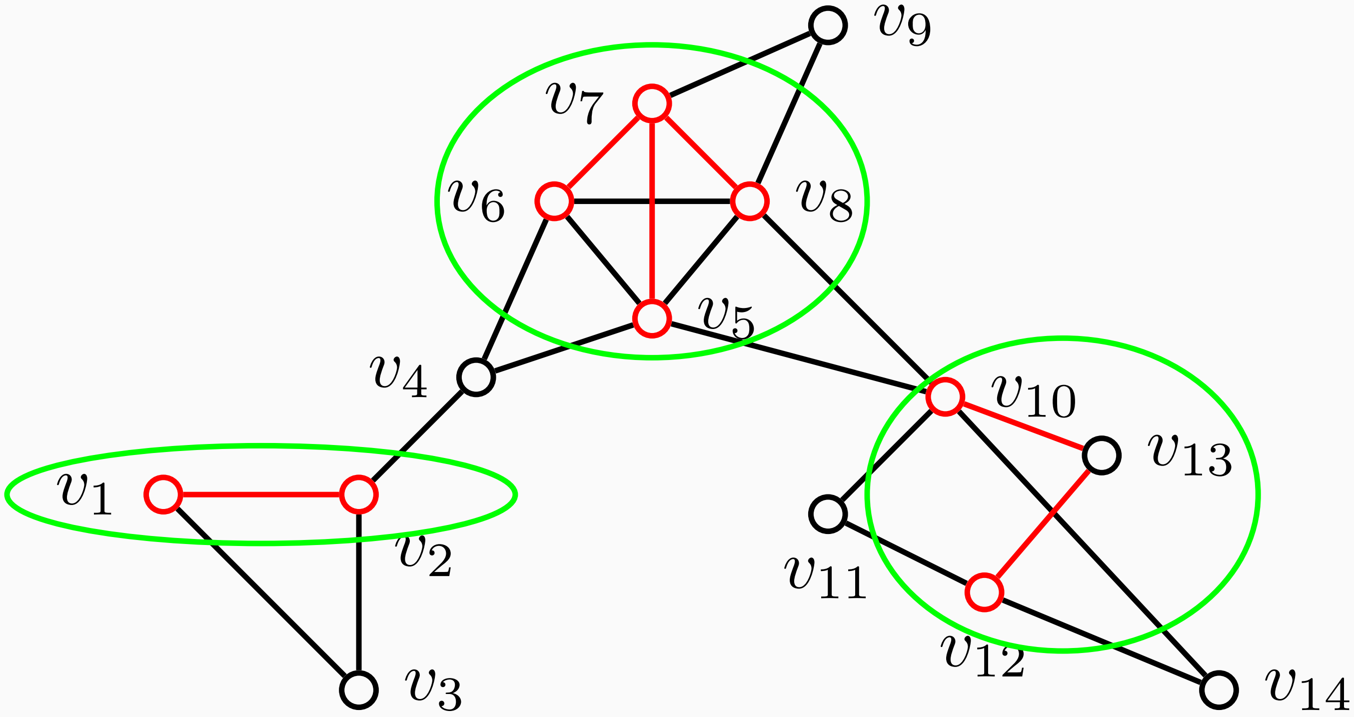
Tree Contraction



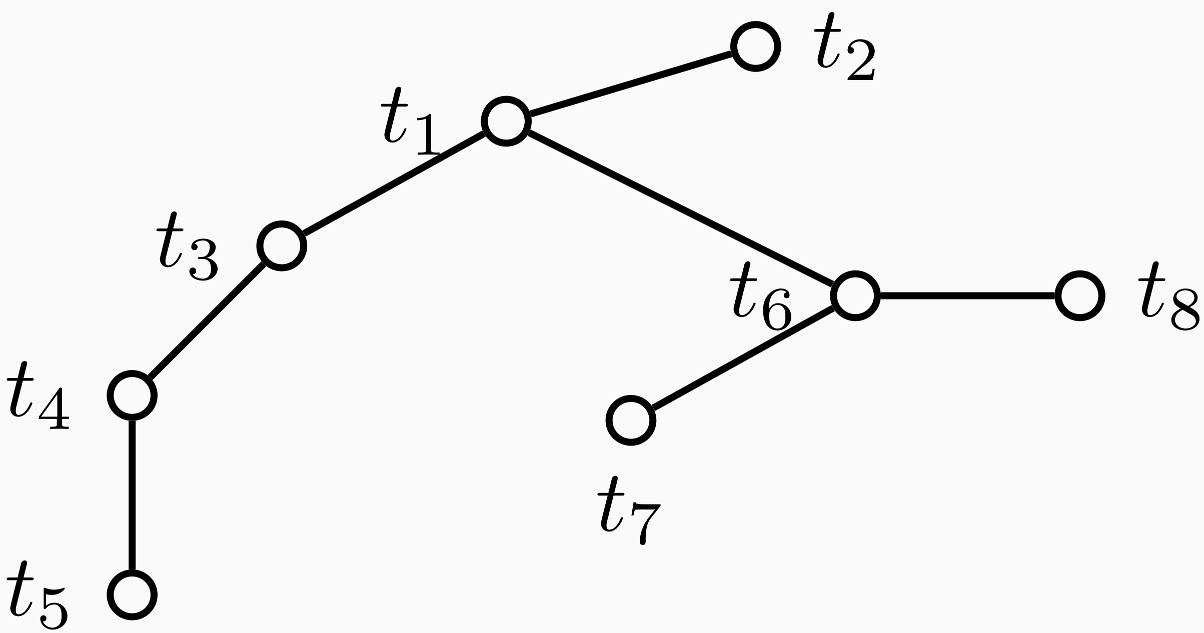
Tree Contraction



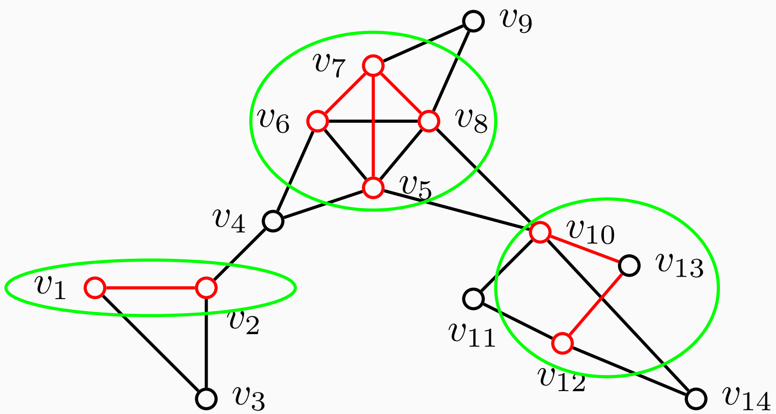
Tree Contraction

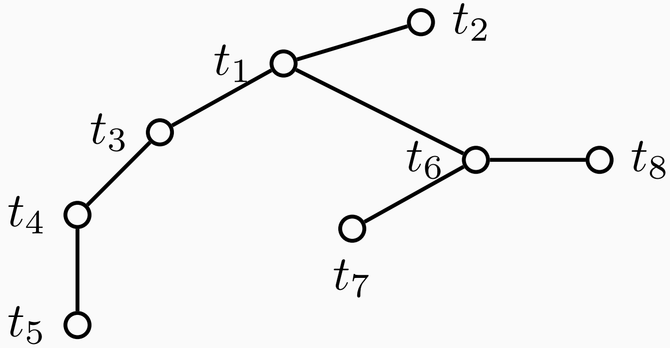


Tree Contraction



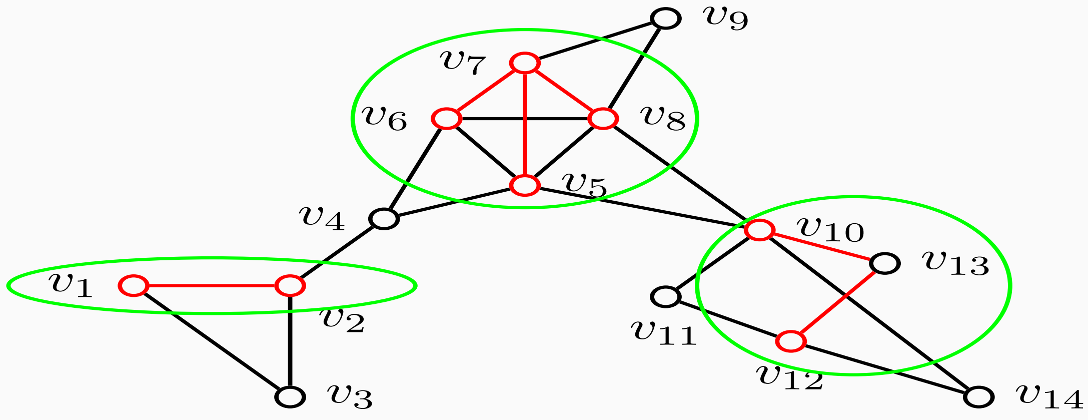
Tree Contraction



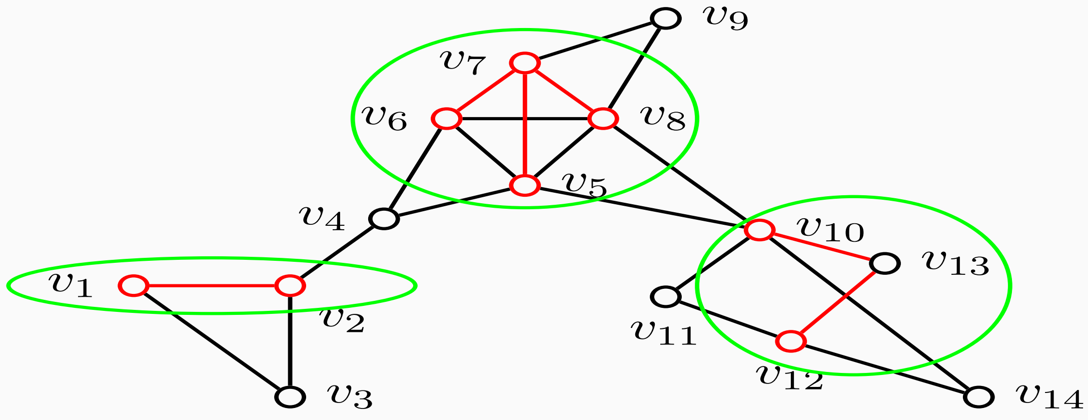


Contraction as a Partition Problem

Contraction as a Partition Problem

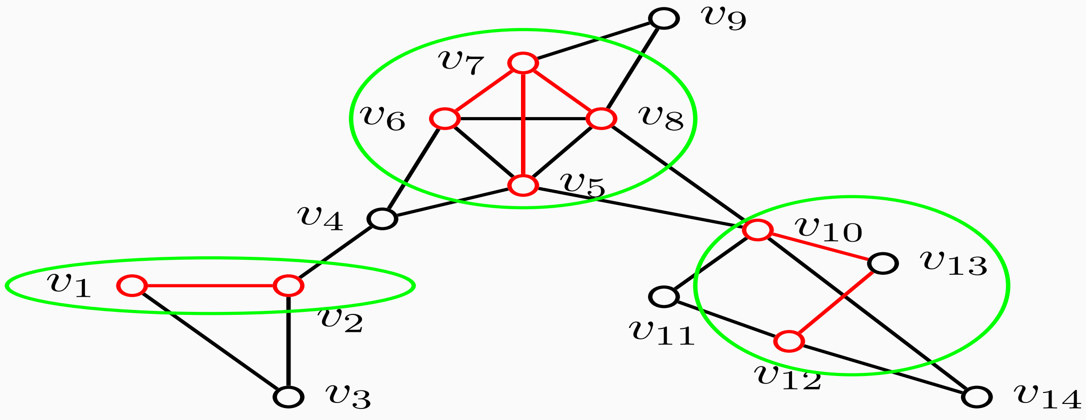


Contraction as a Partition Problem



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.

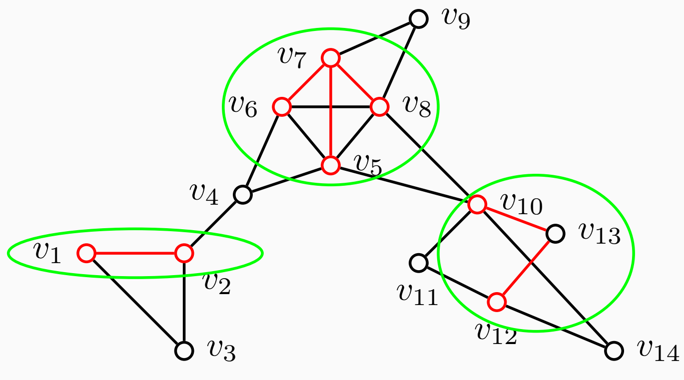
Contraction as a Partition Problem

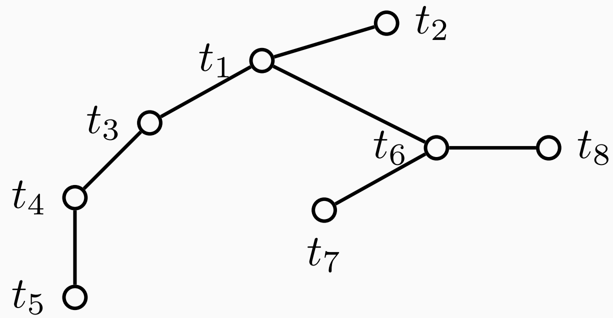


G is contractible to T if there exists a partition of $V(G)$ into $W(t_1), W(t_2), \ldots W(t_{|V(T)|})$ s.t.

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

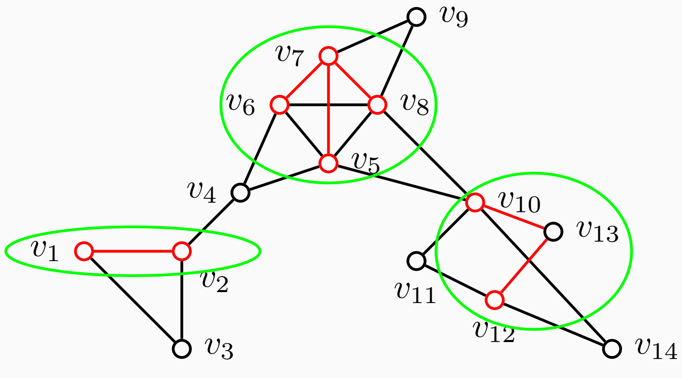
Witness Structure : Definition

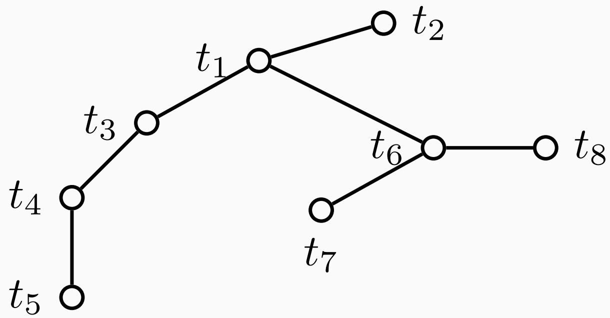




- $\mathcal{W} = \{W(t) \mid t \in V(T)\}$ is called the T-*witness structure* of G

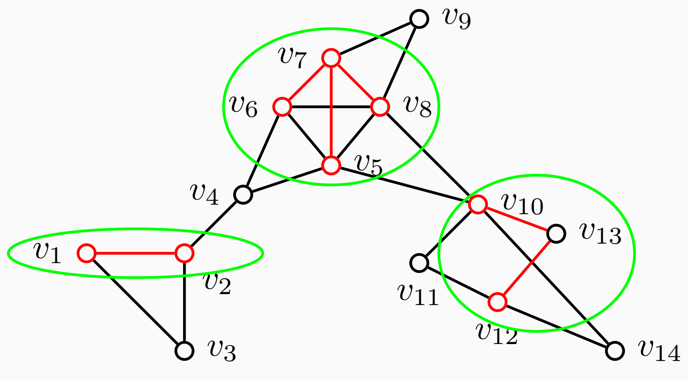
Witness Structure : Definition

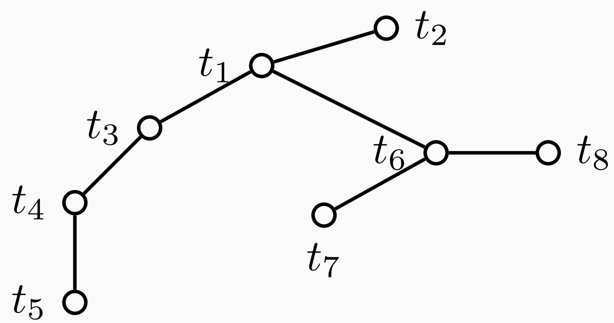




- $\mathcal{W} = \{W(t) \mid t \in V(T)\}$ is called the *T-witness structure* of G
- *Big-witness set* if $|W(t)| > 1$ e.g. $W(t_1), W(t_6), W(t_4)$

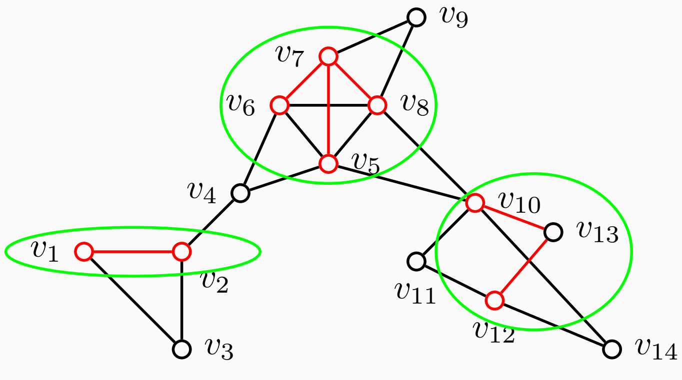
Witness Structure : Definition

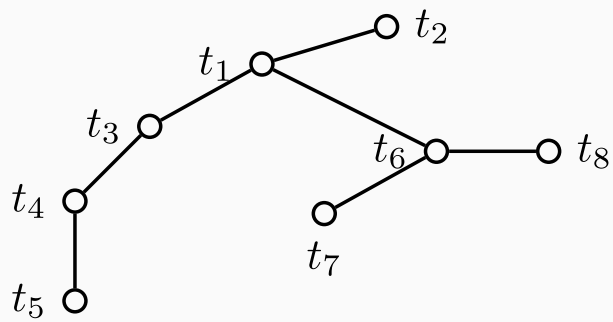




- $\mathcal{W} = \{W(t) \mid t \in V(T)\}$ is called the T-*witness structure* of G
- *Big-witness set* if $|W(t)| > 1$ e.g. $W(t_1), W(t_6), W(t_4)$
- $k = \sum_{t \in V(T)}(|W(t)| - 1)$
  We say G is k-contractible to graph T

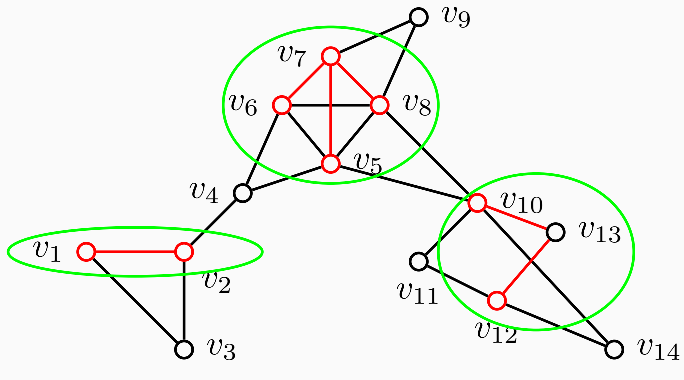
Witness Structure : Observations

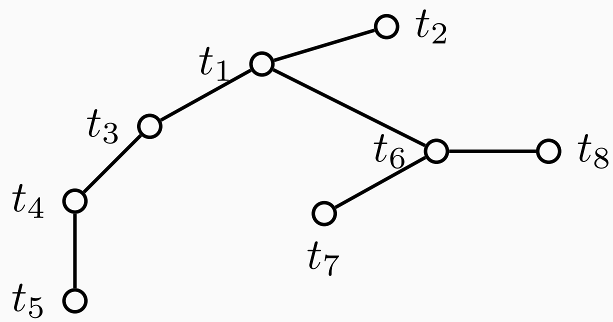




If G is k-contractible to T and $\mathcal{W}$ be its T-witness structure then,

Witness Structure : Observations

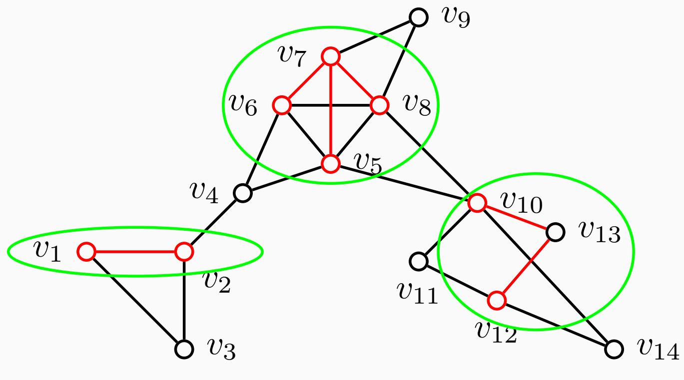


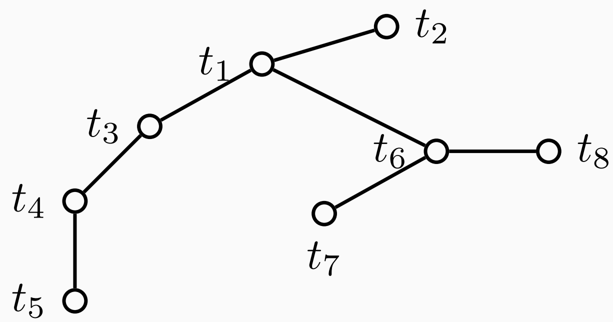


If G is k-contractible to T and $\mathcal{W}$ be its T-witness structure then,

- No witness set in $\mathcal{W}$ contains more than $k+1$ vertices;

Witness Structure : Observations

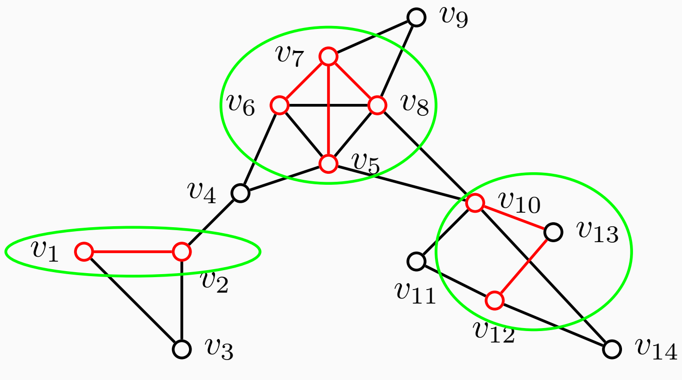


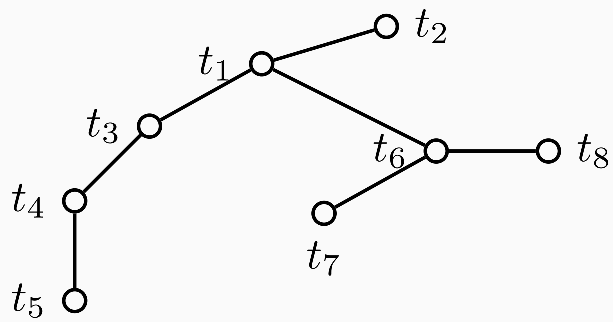


If G is k-contractible to T and $\mathcal{W}$ be its T-witness structure then,

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

Witness Structure : Observations

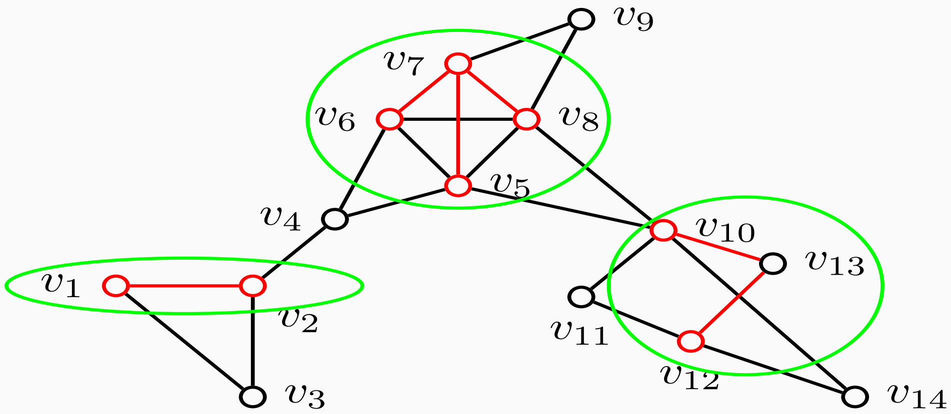




If G is k-contractible to T and $\mathcal{W}$ be its T-witness structure then,

- 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.

Contraction as a Partition Problem



Contraction Problem

- Identify a partition
- Provide connectivity

Lossy Kernelization (Informal Intro)

Kernelization (Reduction Rule)

An algorithm $\mathcal{A}_{Red}$ running in poly(n)

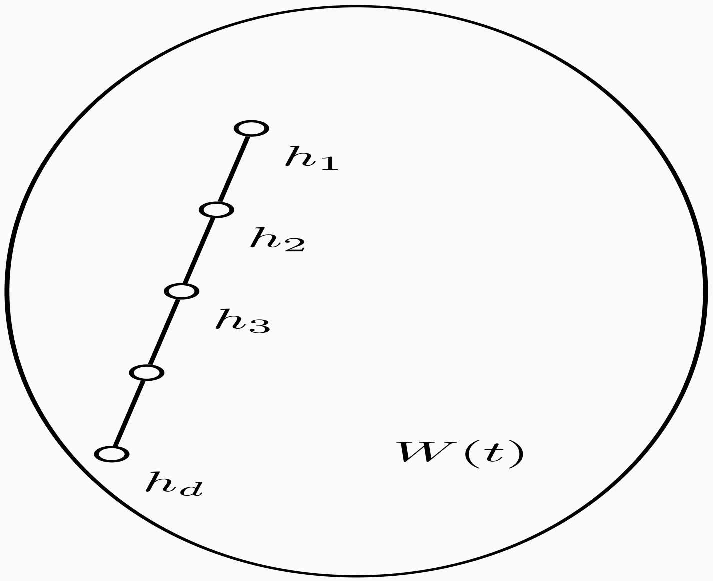
$$(I, k) \longrightarrow (I', k')$$

(I, k) is a yes instance iff (I', k') is a yes instance

Kernelization

In time *poly(n)*, we can find vertices $h_1, h_2, \dots, h_d$ s.t.

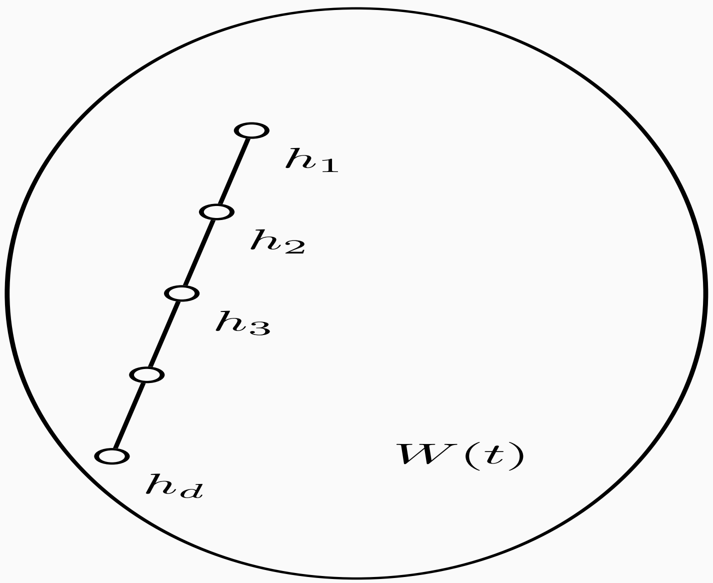
- all these vertices are in one witness set, say $W(t)$, for any optimal solution
- graph induced on these vertices is connected



Kernelization

Algorithm $\mathcal{A}_{Red}$

- In input graph G, find vertices $h_1, h_2, \ldots, h_d$
- Construct G' from G by contracting graph induced on $\{h_1, h_2, \ldots h_d\}$
- Output: $(G', k - (d - 1))$



Lossy Kernelization (Reduction Rule)

A reduction algorithm $\mathcal{A}_{Red}$ running in poly(n)

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

~~(I, k) is a yes instance iff (I', k') is a yes instance~~

Lossy Kernelization (Reduction Rule)

A reduction algorithm $\mathcal{A}_{Red}$ running in poly(n)

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

~~(I, k) is a yes instance iff (I', k') is a yes instance~~

A solution lifting algorithm $\mathcal{A}_{Sol-Lift}$ running in poly(n)

$$\text{Solution } S' \text{ for } (I', k') \longrightarrow \text{ Solution } S \text{ for } (I, k)$$

Lossy Kernelization (Reduction Rule)

A reduction algorithm $\mathcal{A}_{Red}$ running in poly(n)

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

~~(I, k) is a yes instance iff (I', k') is a yes instance~~

A solution lifting algorithm $\mathcal{A}_{Sol-Lift}$ running in poly(n)

$$\text{Solution } S' \text{ for } (I', k') \longrightarrow \text{ Solution } S \text{ for } (I, k)$$

such that solution S to (I, k) is as good as solution S' was to (I', k').

Lossy Kernelization (Reduction Rule)

A reduction algorithm $\mathcal{A}_{Red}$ running in poly(n)

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

~~(I, k) is a yes instance iff (I', k') is a yes instance~~

A solution lifting algorithm $\mathcal{A}_{Sol-Lift}$ running in poly(n)

$$\text{Solution } S' \text{ for } (I', k') \longrightarrow \text{ Solution } S \text{ for } (I, k)$$

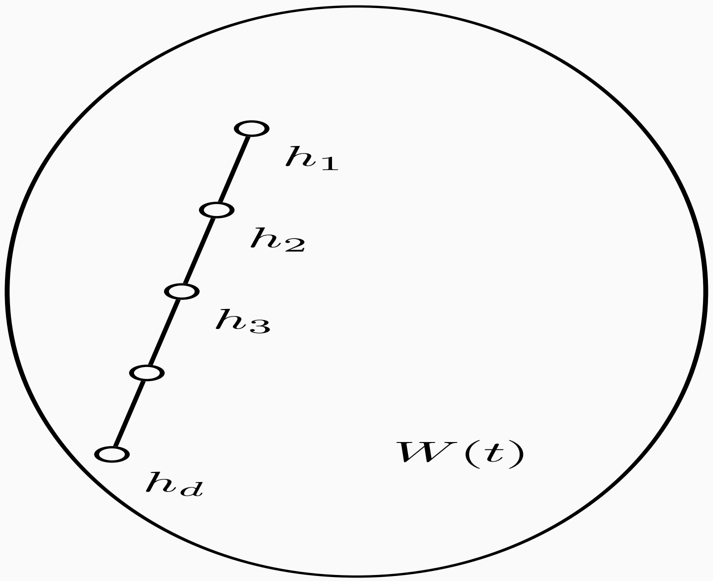
such that solution S to (I, k) is as good as solution S' was to (I', k').

$\mathcal{A}_{Sol-Lift}$ have access to $\mathcal{A}_{Red}$

Lossy Kernelization

In time *poly(n)*, we can find vertices $h_1, h_2, \ldots, h_d$ s.t.

- all these vertices are in one witness set, say $W(t)$, for any optimal solution
- graph induced on these vertices is connected



Lossy Kernelization

Algorithm $\mathcal{A}_{Red}$

- In input graph G, find vertices $h_1, h_2, \ldots, h_d$
- Construct G' from G by contracting graph induced on $\{h_1, h_2, \ldots h_d\}$. (Let F' be set of contracted edges.)
- Output: $(G', k - (d - 1))$

Lossy Kernelization

Algorithm $\mathcal{A}_{Red}$

- In input graph G, find vertices $h_1, h_2, \ldots, h_d$
- Construct G' from G by contracting graph induced on $\{h_1, h_2, \ldots h_d\}$. (Let F' be set of contracted edges.)
- Output: $(G', k - (d - 1))$

Algorithm $\mathcal{A}_{Sol-Lift}$

- Input : S' a solution to (G', k'); (G', k'); (G, k) (and hence F')
- Output: $S' \cup F'$

Lossy Kernelization

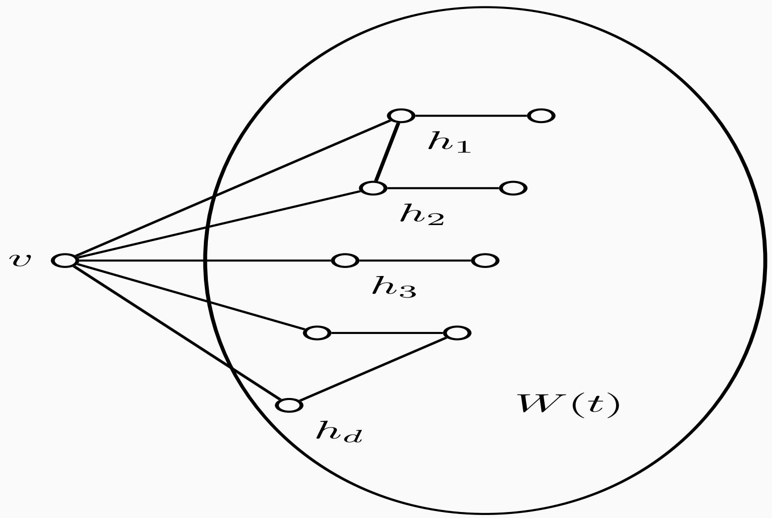
In time *poly(n)*, we can find vertices $h_1, h_2, \ldots, h_d$ s.t.

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

Lossy Kernelization

In time *poly(n)*, we can find vertices $h_1, h_2, \ldots, h_d$ s.t.

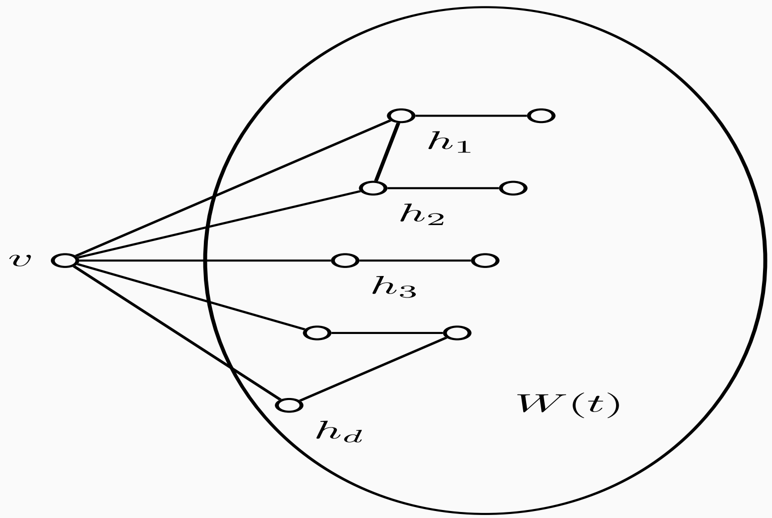
- all these vertices are in one witness set, say $W(t)$, for any optimal solution
- there exists v such that $\{h_1, h_2, \ldots, h_d\} \subseteq N(v)$



Lossy Kernelization

In time *poly(n)*, we can find vertices $h_1, h_2, \ldots, h_d$ s.t.

- all these vertices are in one witness set, say $W(t)$, for any optimal solution
- 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

We have not found entire $W(t)$; v may or may not be in $W(t)$.

Lossy Kernelization

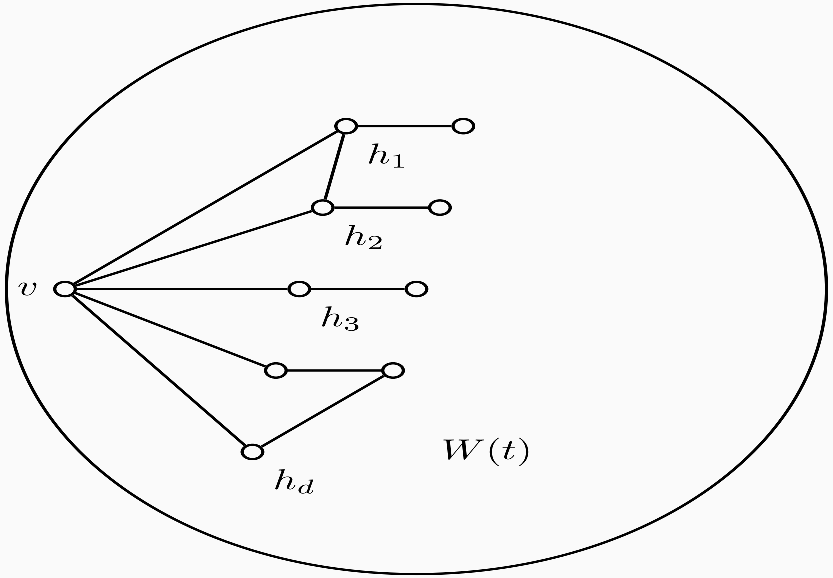
We have not found entire $W(t)$; v may or may not be in $W(t)$.

Introducing lossy-ness : Add vertex v to $W(t)$ for connectivity

Lossy Kernelization

We have not found entire $W(t)$; v may or may not be in $W(t)$.

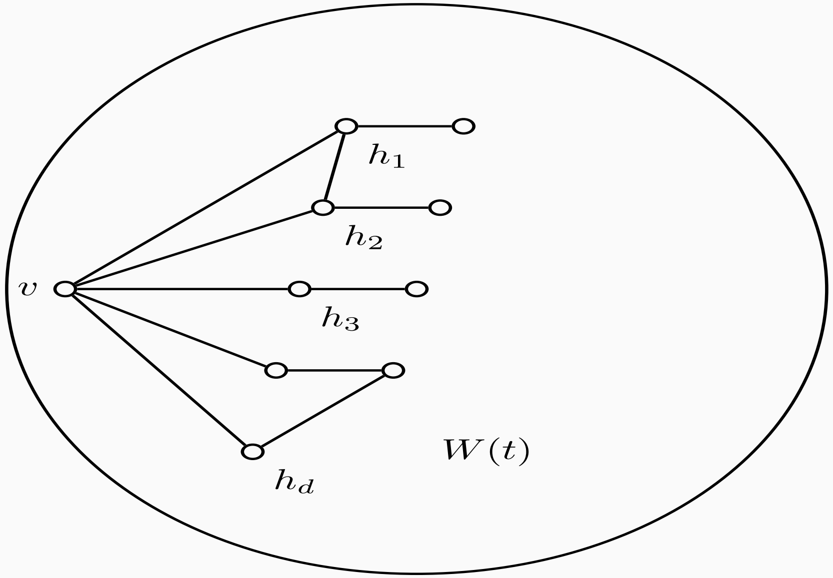
Introducing lossy-ness : Add vertex v to $W(t)$ for connectivity



Lossy Kernelization

We have not found entire $W(t)$; v may or may not be in $W(t)$.

Introducing lossy-ness : Add vertex v to $W(t)$ for connectivity

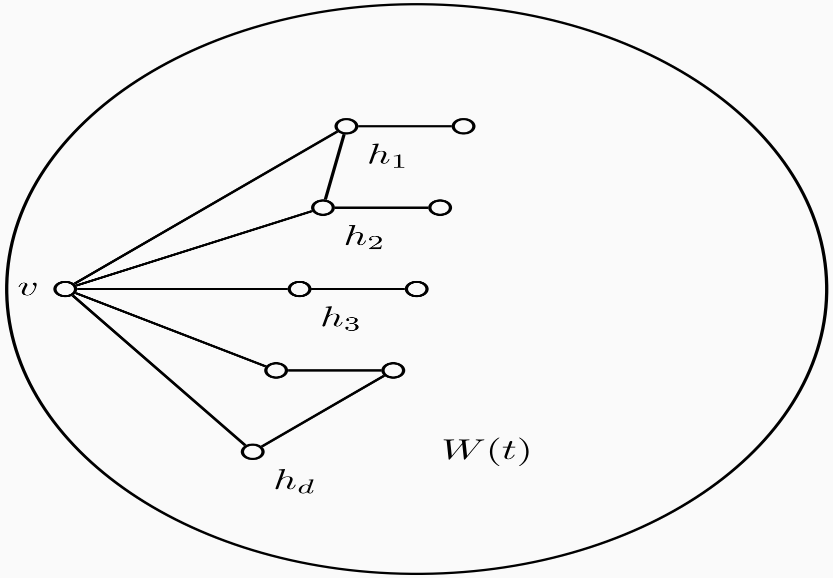


Contract all edges $\{vh_i | \forall i \in [d]\}$ to get new instance $(G', k - (d-1))$

Lossy Kernelization

We have not found entire $W(t)$; v may or may not be in $W(t)$.

Introducing lossy-ness : Add vertex v to $W(t)$ for connectivity



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

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

Lossy Kernelization

Algorithm $\mathcal{A}_{Red}$

- In input graph G, find vertices $h_1, h_2, \ldots, h_d$ & v_1
- Construct G' from G by contracting graph induced on $\{h_1, h_2, \ldots h_d, v_1\}$. (Let F' be set of contracted edges.)
- Output: $(G', k - (d - 1))$

Lossy Kernelization

Algorithm $\mathcal{A}_{Red}$

- In input graph G, find vertices $h_1, h_2, \ldots, h_d$ & v_1
- Construct G' from G by contracting graph induced on $\{h_1, h_2, \ldots h_d, v_1\}$. (Let F' be set of contracted edges.)
- Output: $(G', k - (d-1))$

Algorithm $\mathcal{A}_{Sol-Lift}$

- Input : S' a solution for (G', k'); (G', k'); (G, k) (and hence F')
- Output: $S' \cup F'$

Lossy Kernelization

Claim: Solution S to (G, k) is as good as solution S' was to (G', k').

Lossy Kernelization

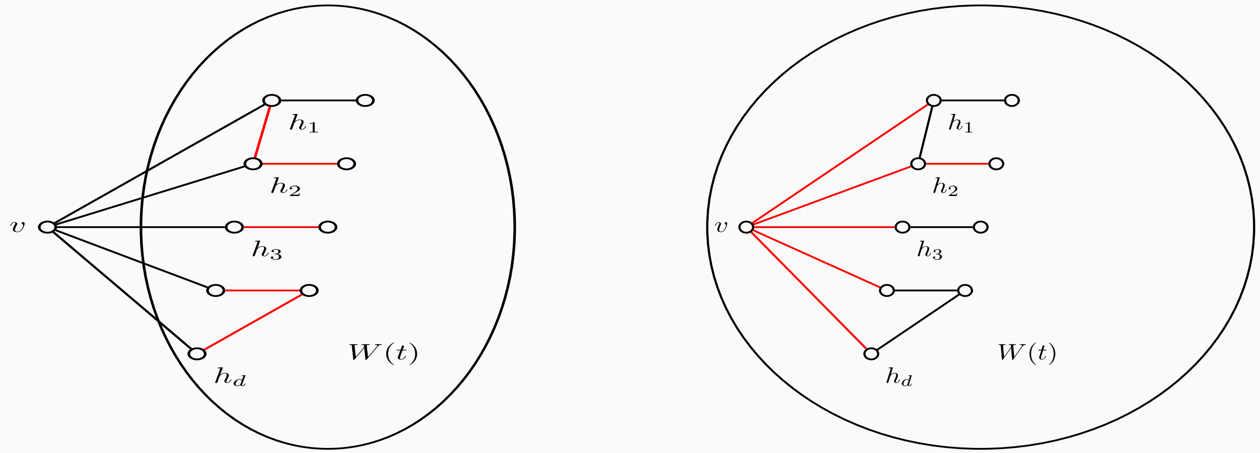
Claim: Solution S to (G, k) is as good as solution S' was to (G', k').

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

Lossy Kernelization

Claim: Solution S to (G, k) is as good as solution S' was to (G', k').

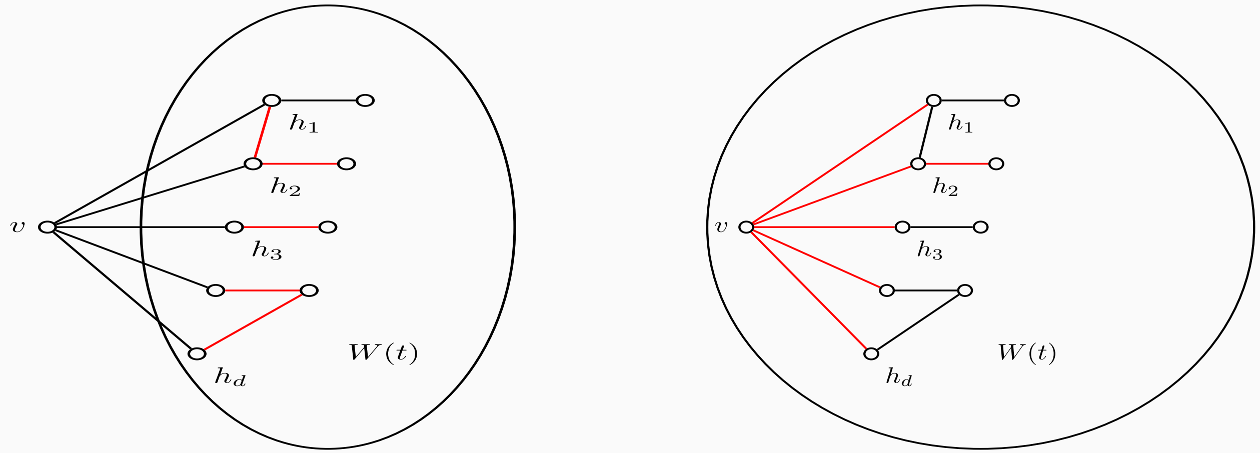
$h_1, h_2, \ldots, h_d$ are in big-witness set $\Rightarrow$ there are $(d-1)$ solution edges incident these vertices



Lossy Kernelization

Claim: Solution S to (G, k) is as good as solution S' was to (G', k').

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

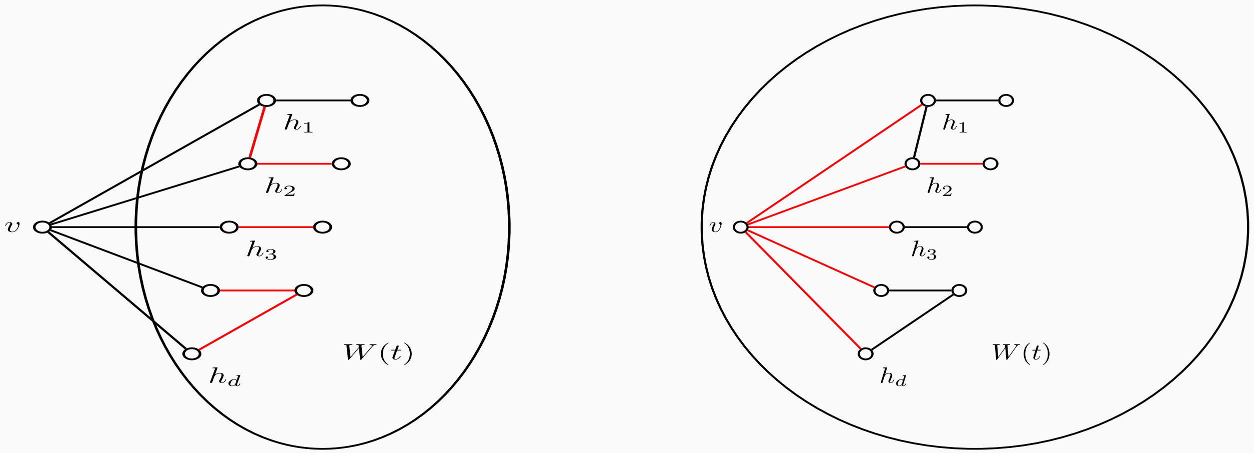


Contracting d-many edges for every $(d-1)$ edges in the solution.

Lossy Kernelization

Claim: Solution S to (G, k) is as good as solution S' was to (G', k').

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



Contracting d-many edges for every $(d-1)$ edges in the solution. The number of edges contracted in this process is $\frac{d}{d-1} = \alpha$ times that of optimum solution

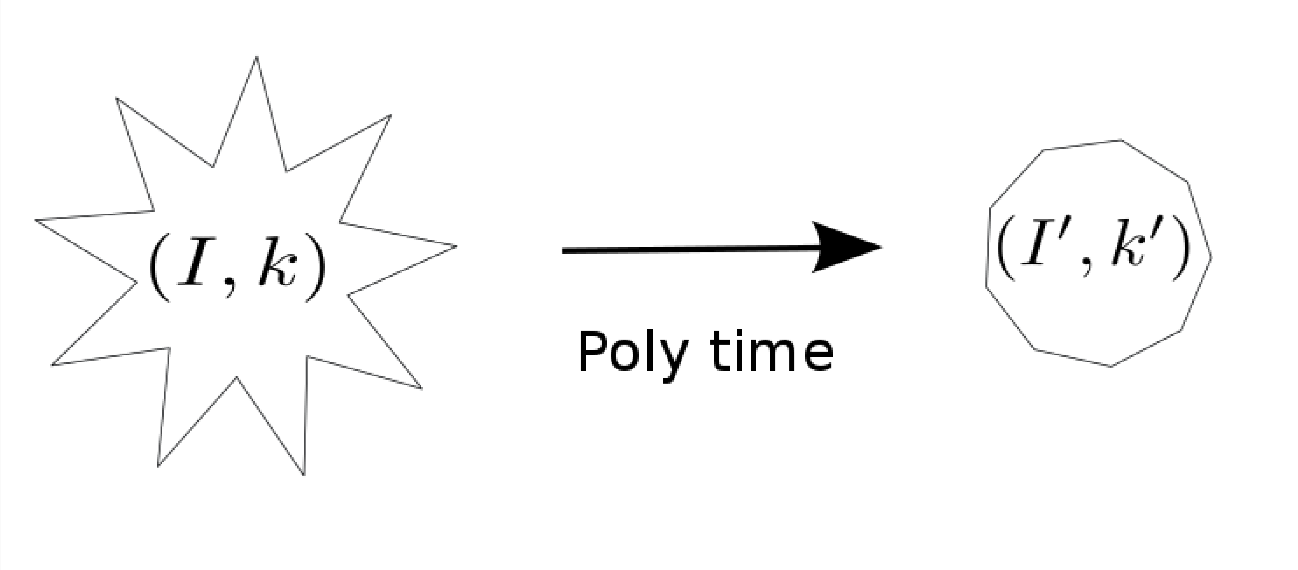
Lossy Kernelization

Claim: Solution S to (G, k) is as good as solution S' was to (G', k').

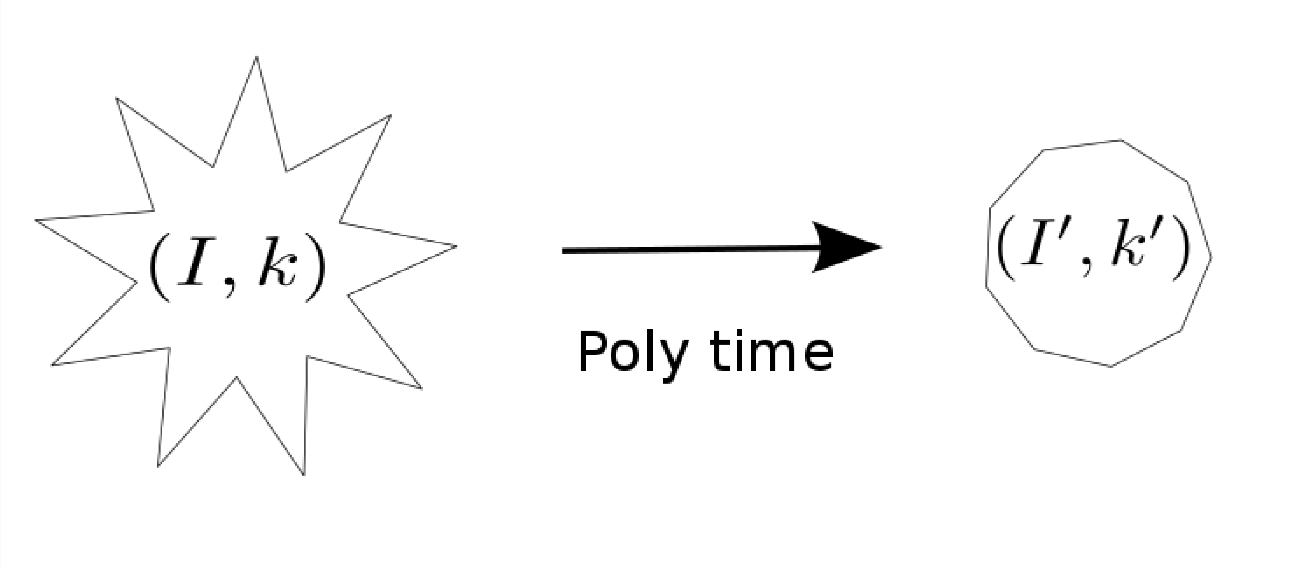
If S' is c-factor approximate solution to (G', k') then S is $\max\{c, \alpha\}$-factor solution to (G, k).

Lossy Kernelization

Kernelization



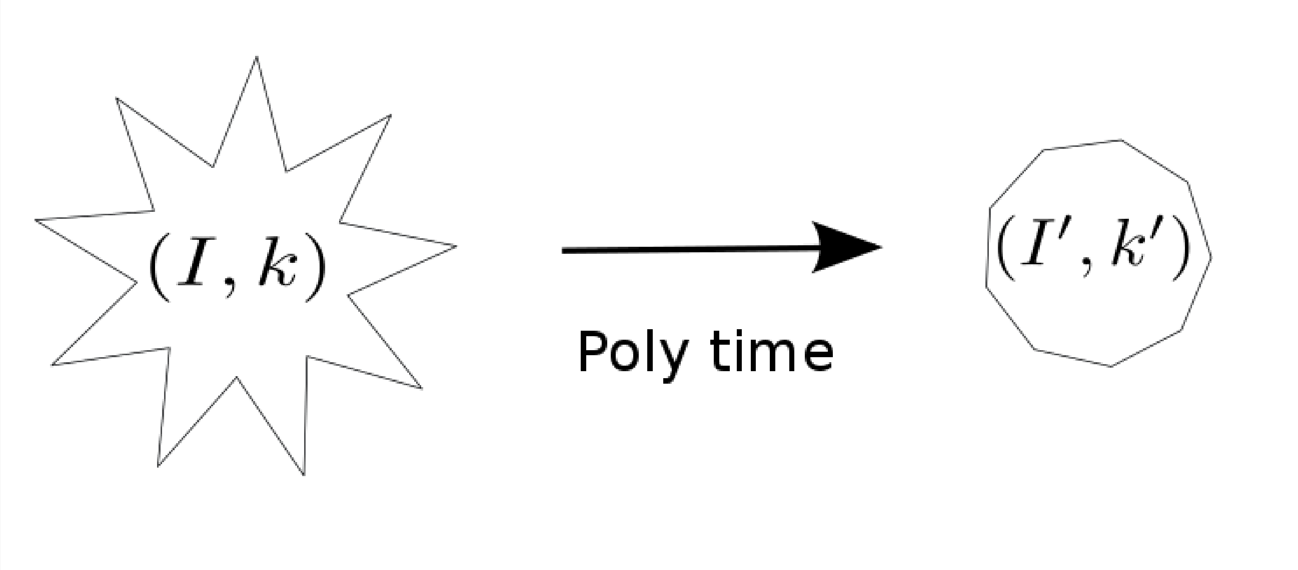
Kernelization



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

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

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

Optimization Version

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

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

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

Optimization Version

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

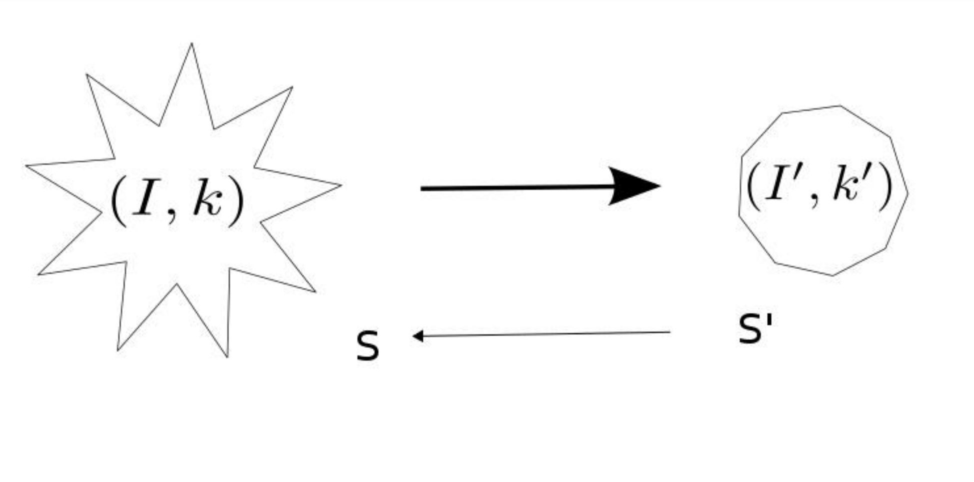
$$\Pi : \Sigma^* \times \mathbb{N} \times \Sigma^* \to \mathbb{R} \cup \{\pm\infty\}$$

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,

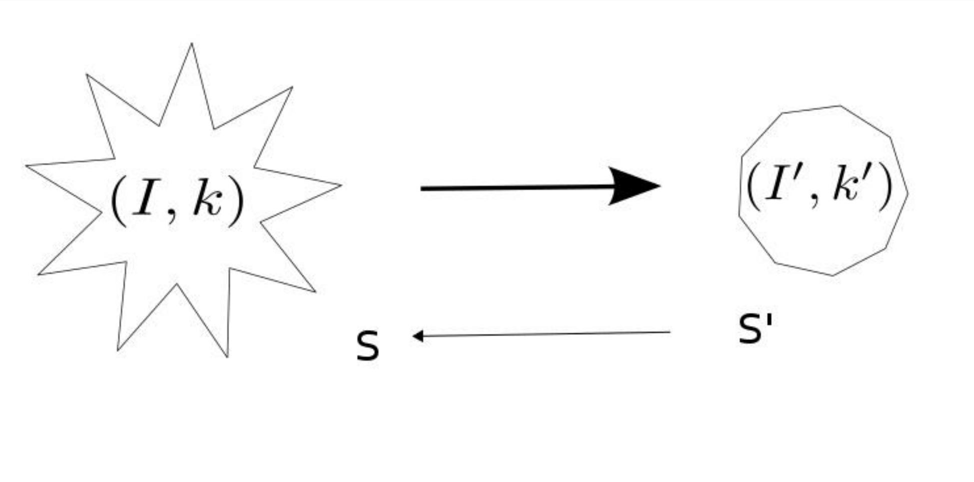
$$\mathrm{OPT}_\Pi(I, k) = \min_{S \in \Sigma^*; |S| \leq |I| + k} \{\Pi(I, k, S)\}$$

Lossy Kernelization



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

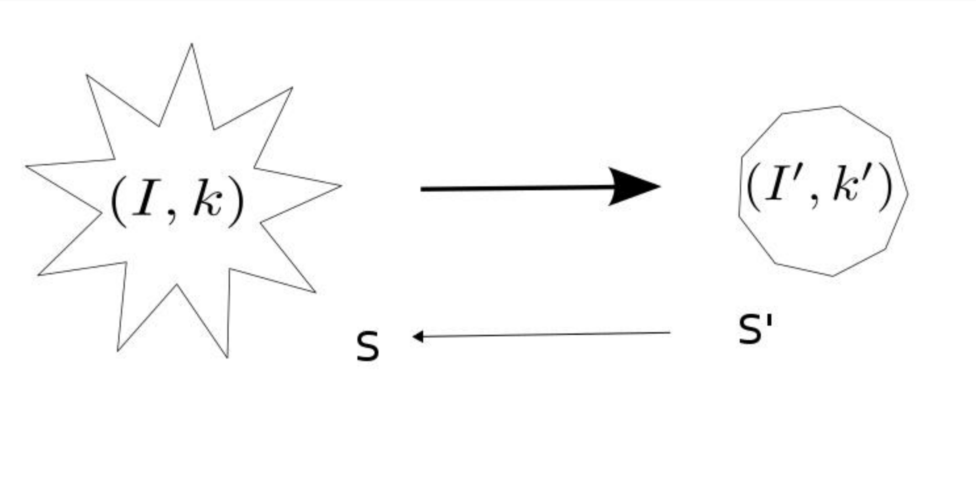
Lossy Kernelization



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

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

Lossy Kernelization



Given (I', k', S') can we construct a solution S to (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?

Lossy Kernelization

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 (Strict 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

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

Lossy Kernelization

Definition (Strict α-approximate kernel)

For a parameterized minimization problem Π if

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

Lossy Kernelization

Minimization Problem

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

Lossy Kernelization

Minimization Problem

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

all solutions of size larger than $k + 1$ are equally bad.

Lossy Kernelization

Minimization Problem

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

all solutions of size larger than $k+1$ are equally bad.

Maximization Problem

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

Lossy Kernelization

Minimization Problem

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

all solutions of size larger than $k+1$ are equally bad.

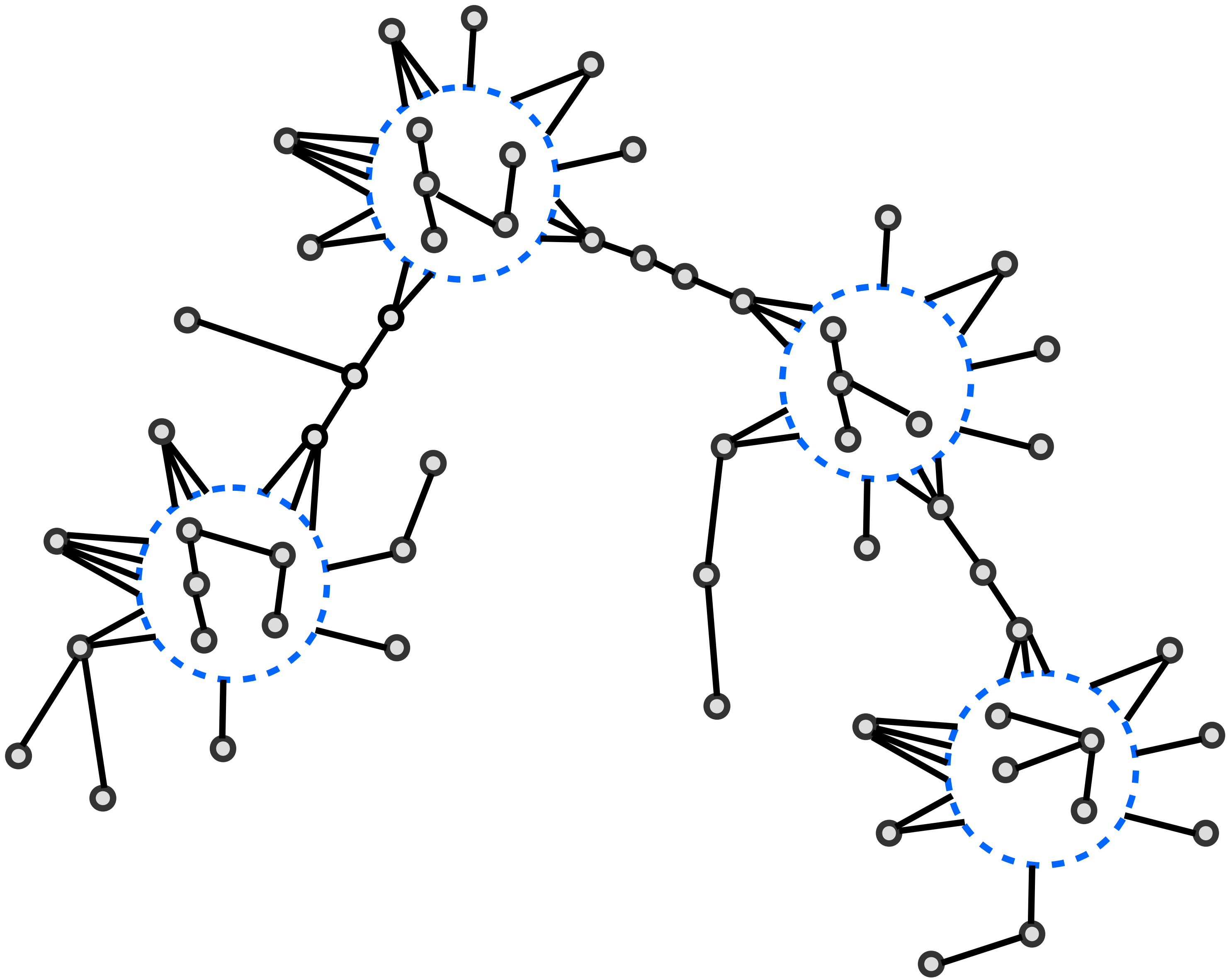
Maximization Problem

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

all solutions of size larger than $k+1$ are equally good.

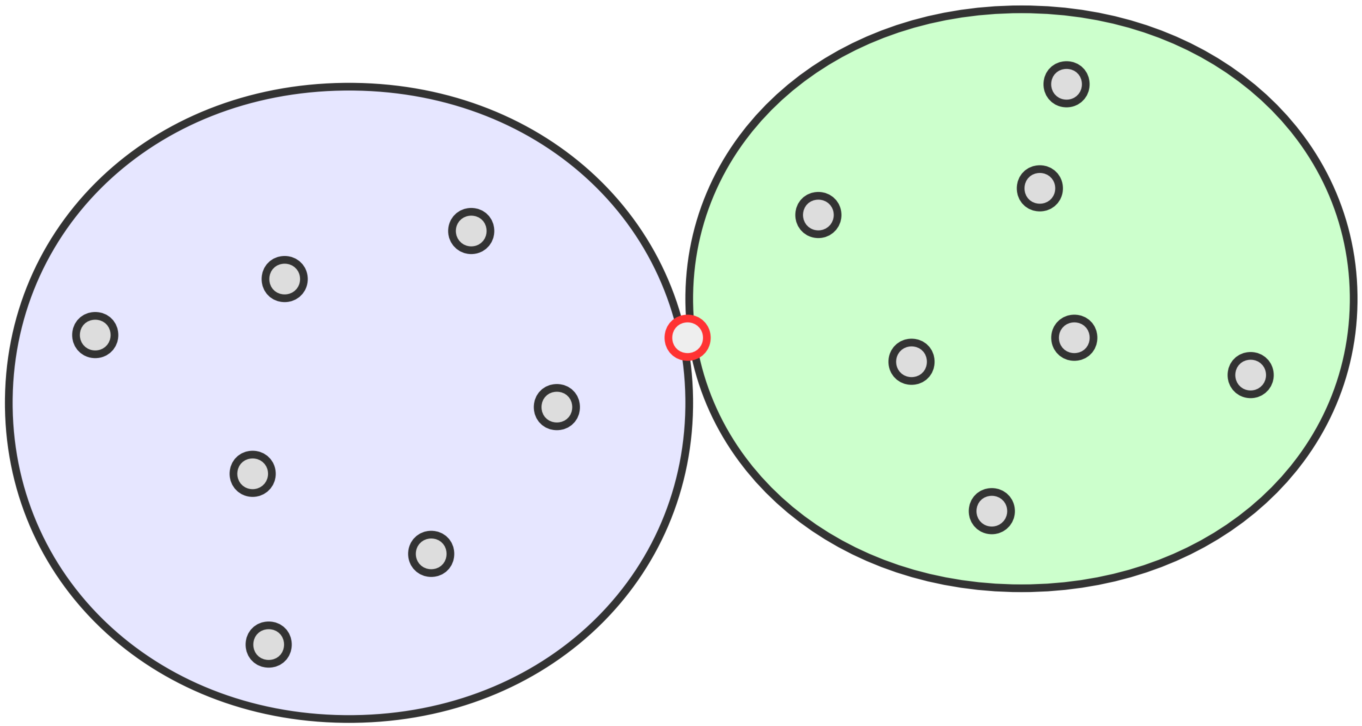
Lossy Kernel for Tree Contraction

Tree Contraction

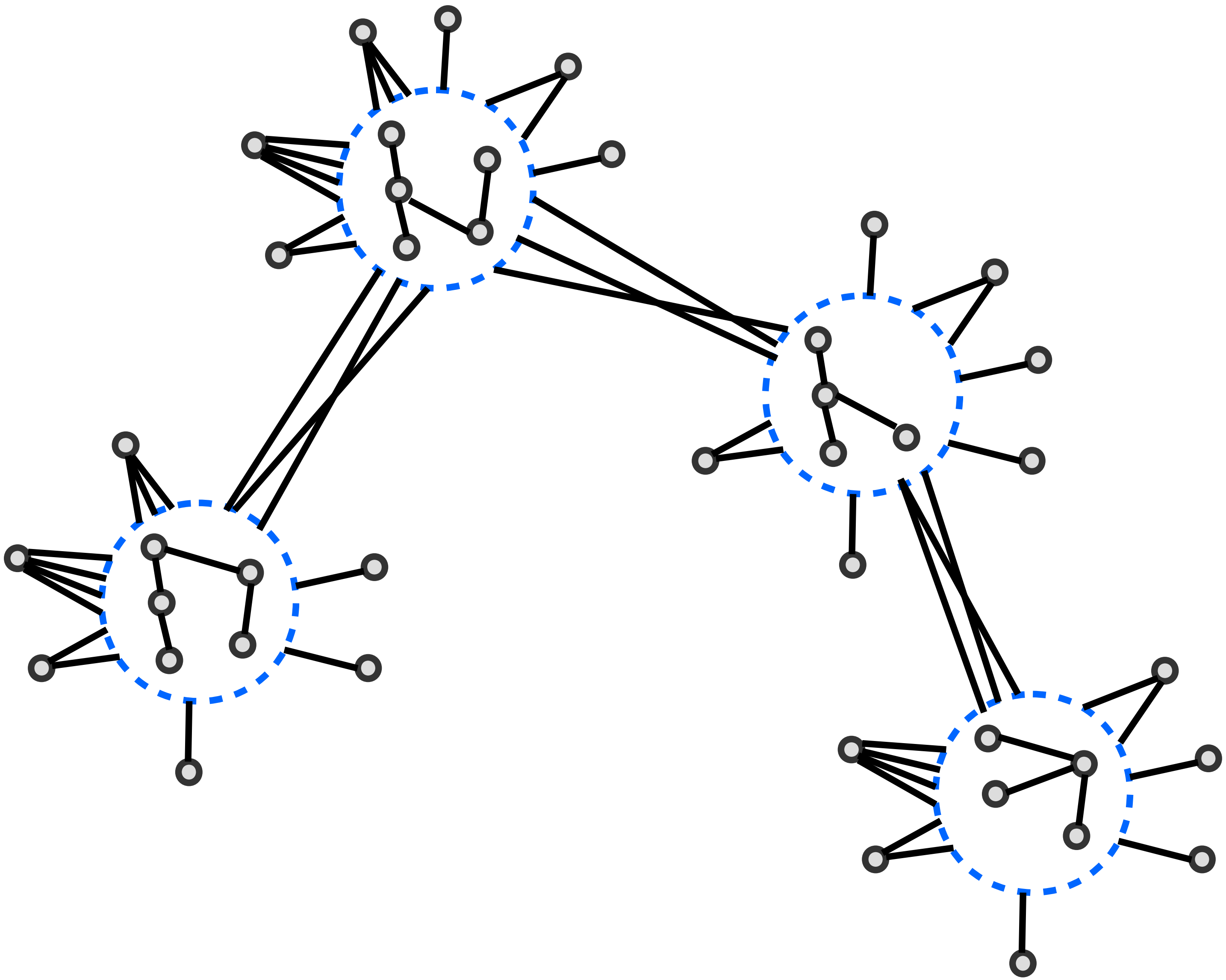


Tree Contraction

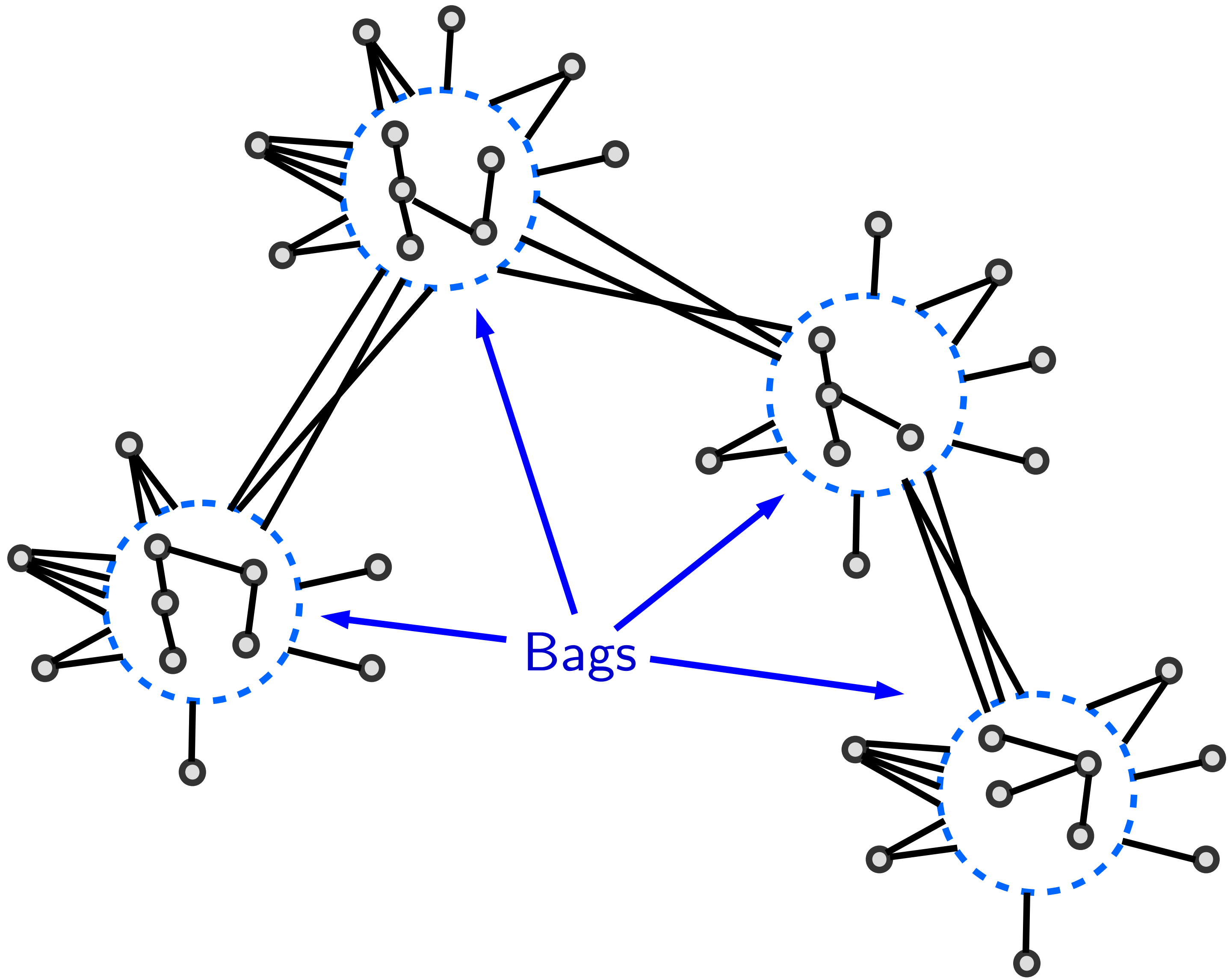
Consider each 2-vertex connected component separately



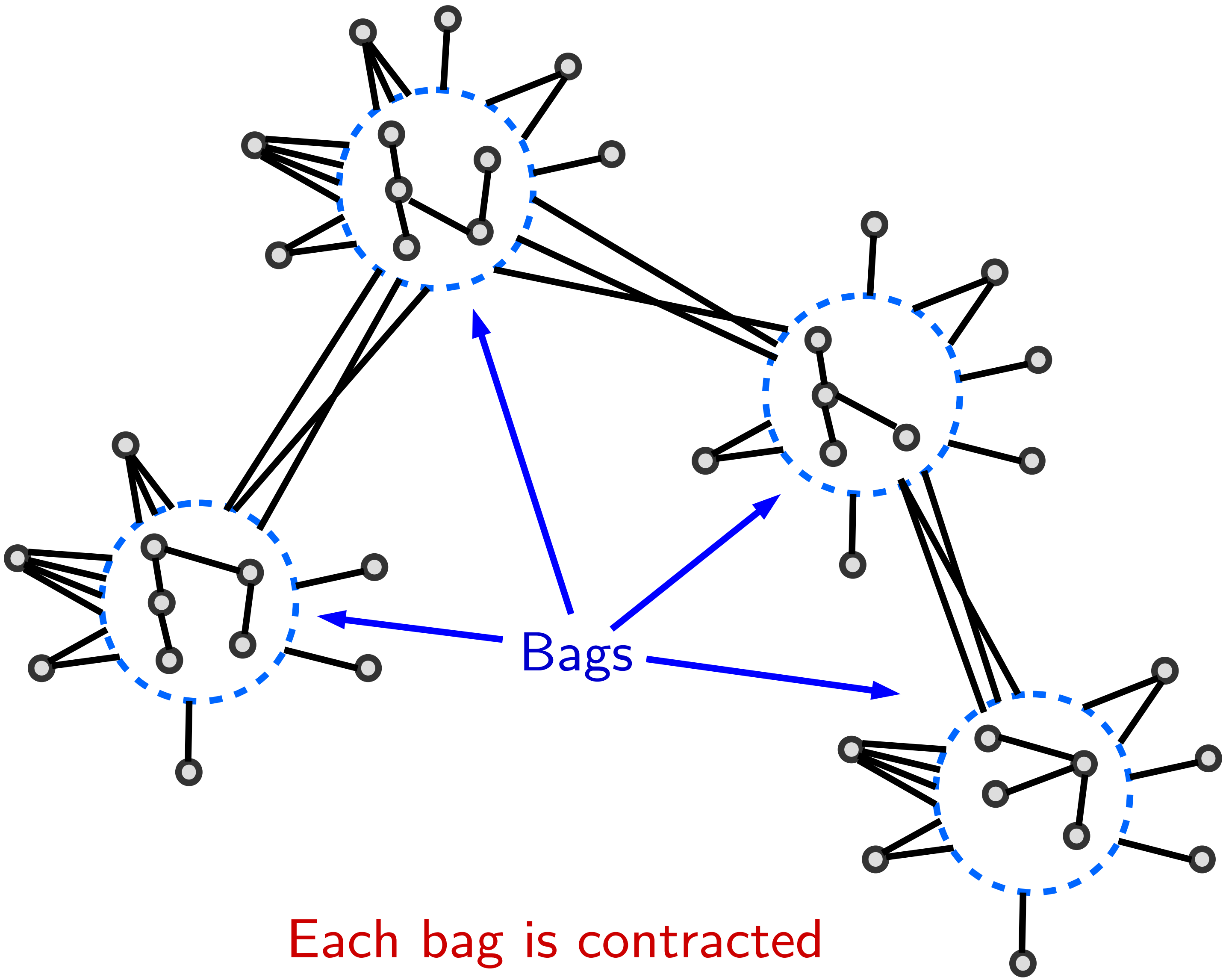
<u>Tree Contraction</u>



<u>Tree Contraction</u>

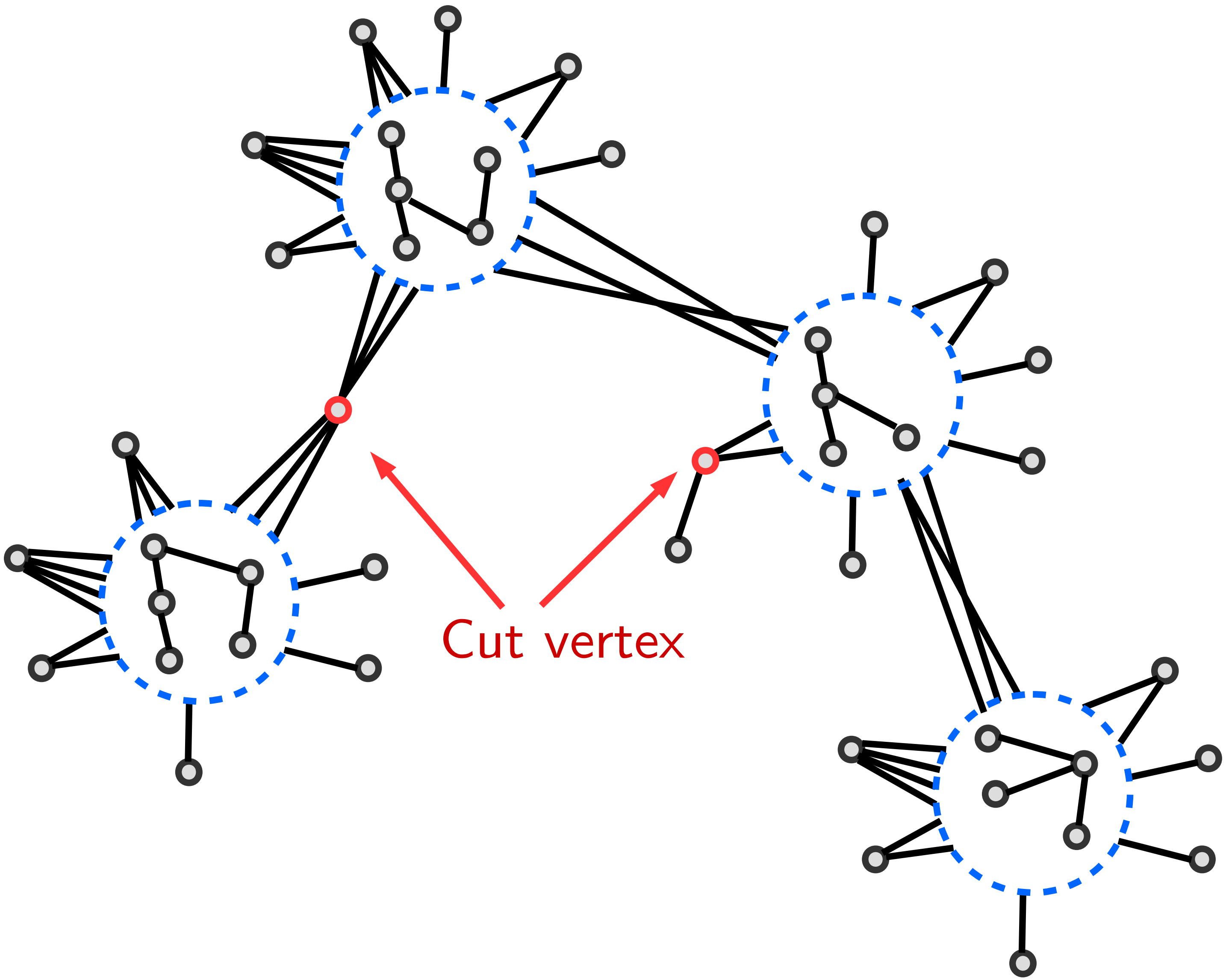


Tree Contraction

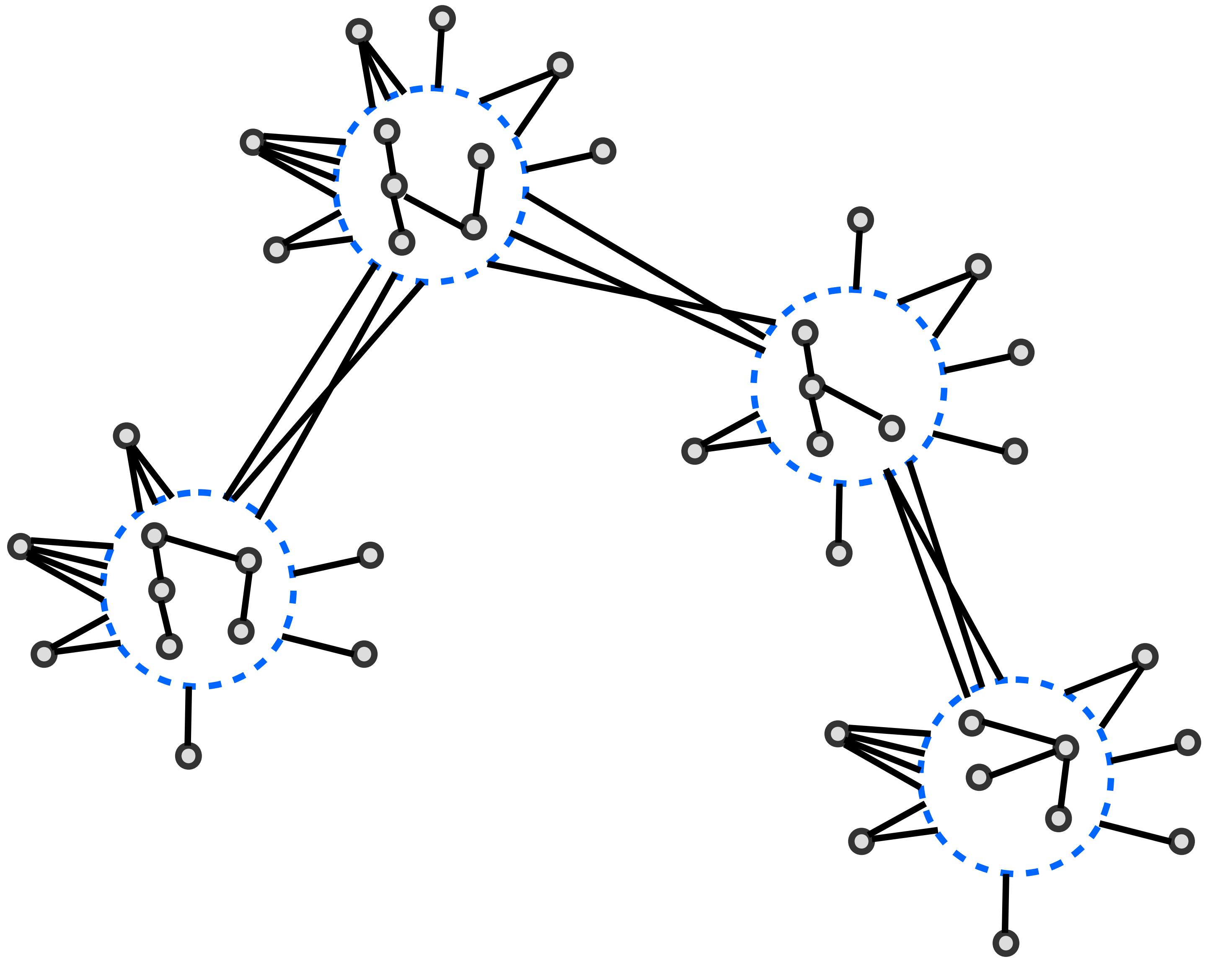


Each bag is contracted
to a distinct vertex

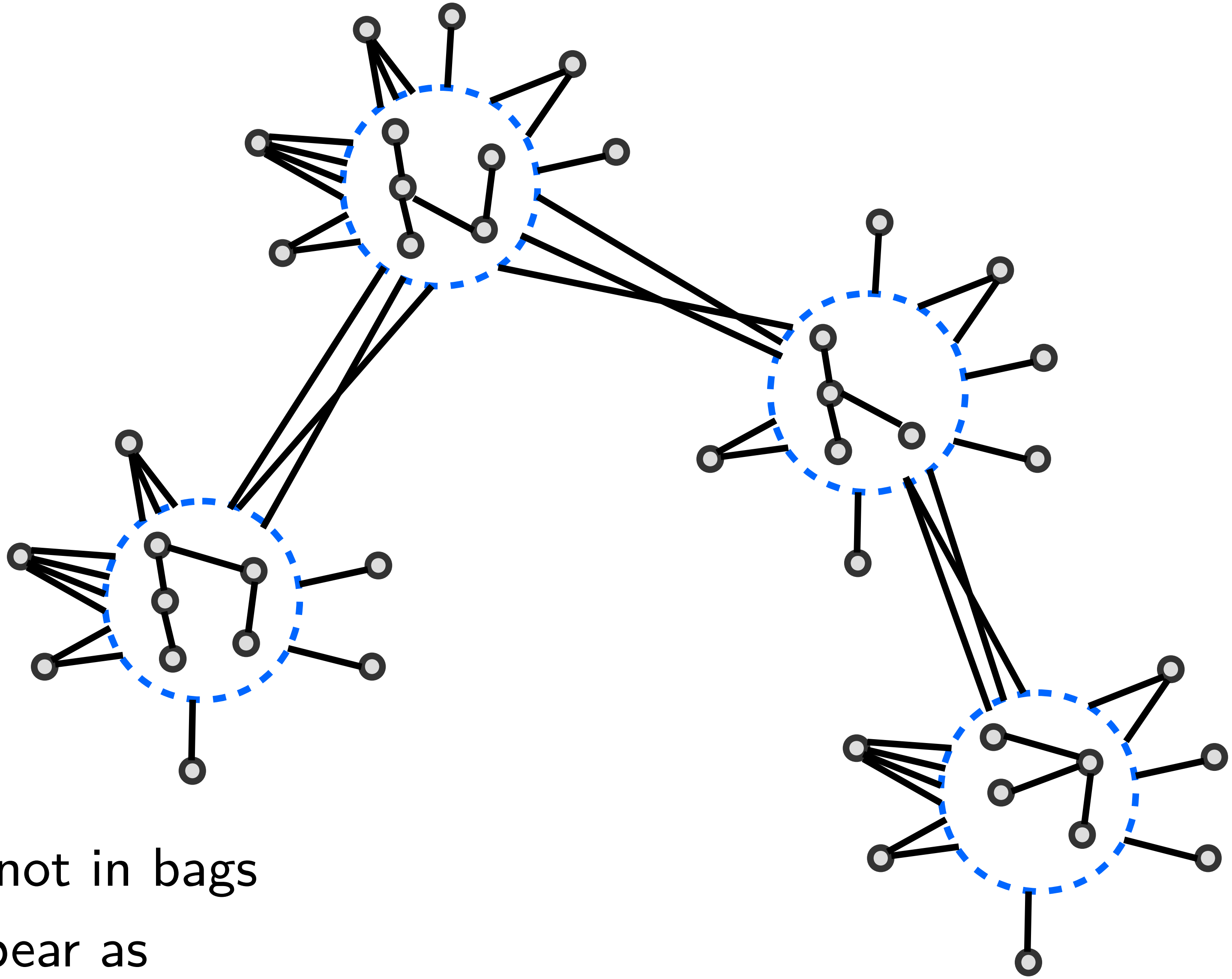
Tree Contraction



Tree Contraction



Tree Contraction

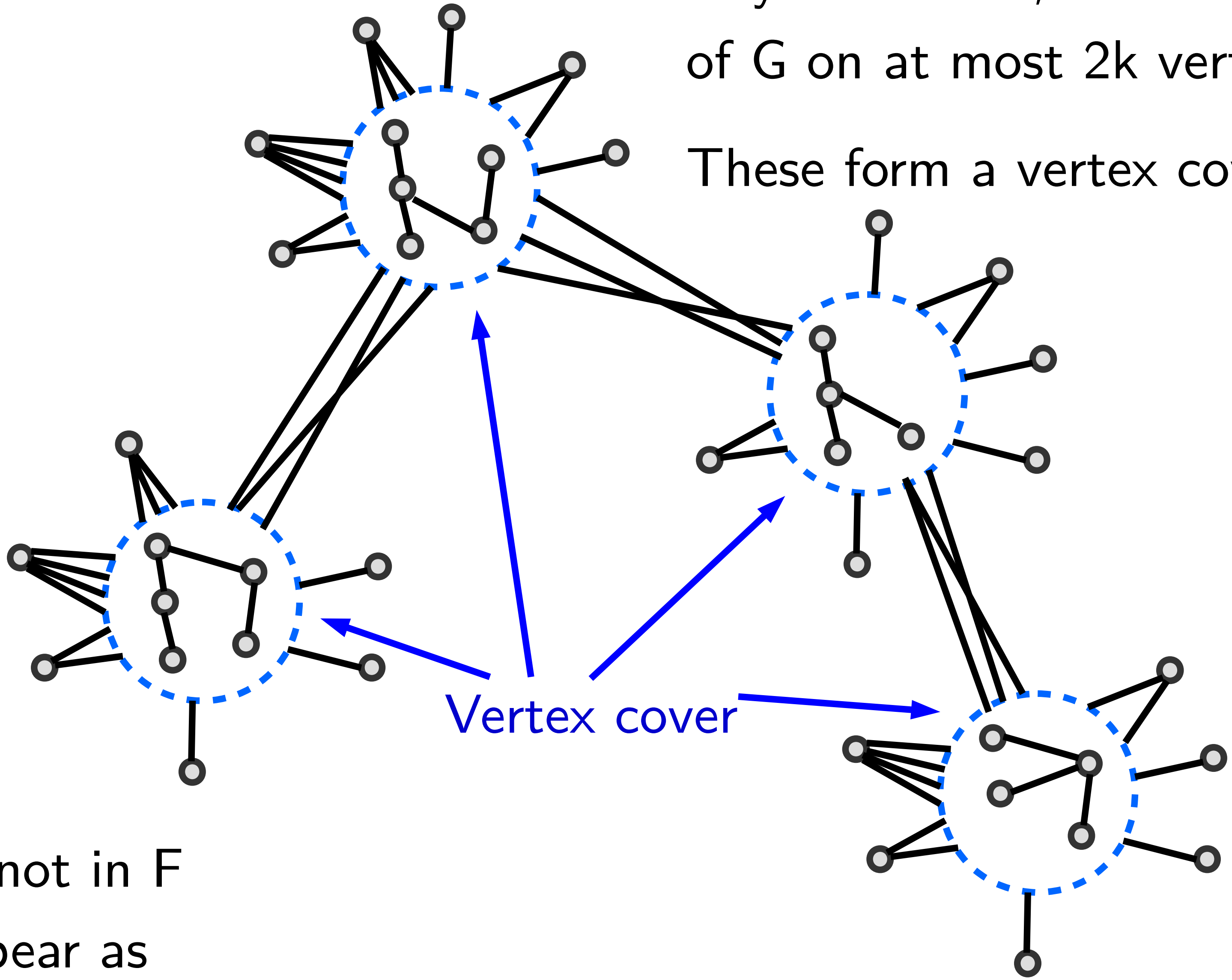


Vertices not in bags
must appear as
“leaves”

Tree Contraction

Any solution S, is a sub-forest of G on at most 2k vertices.

These form a vertex cover

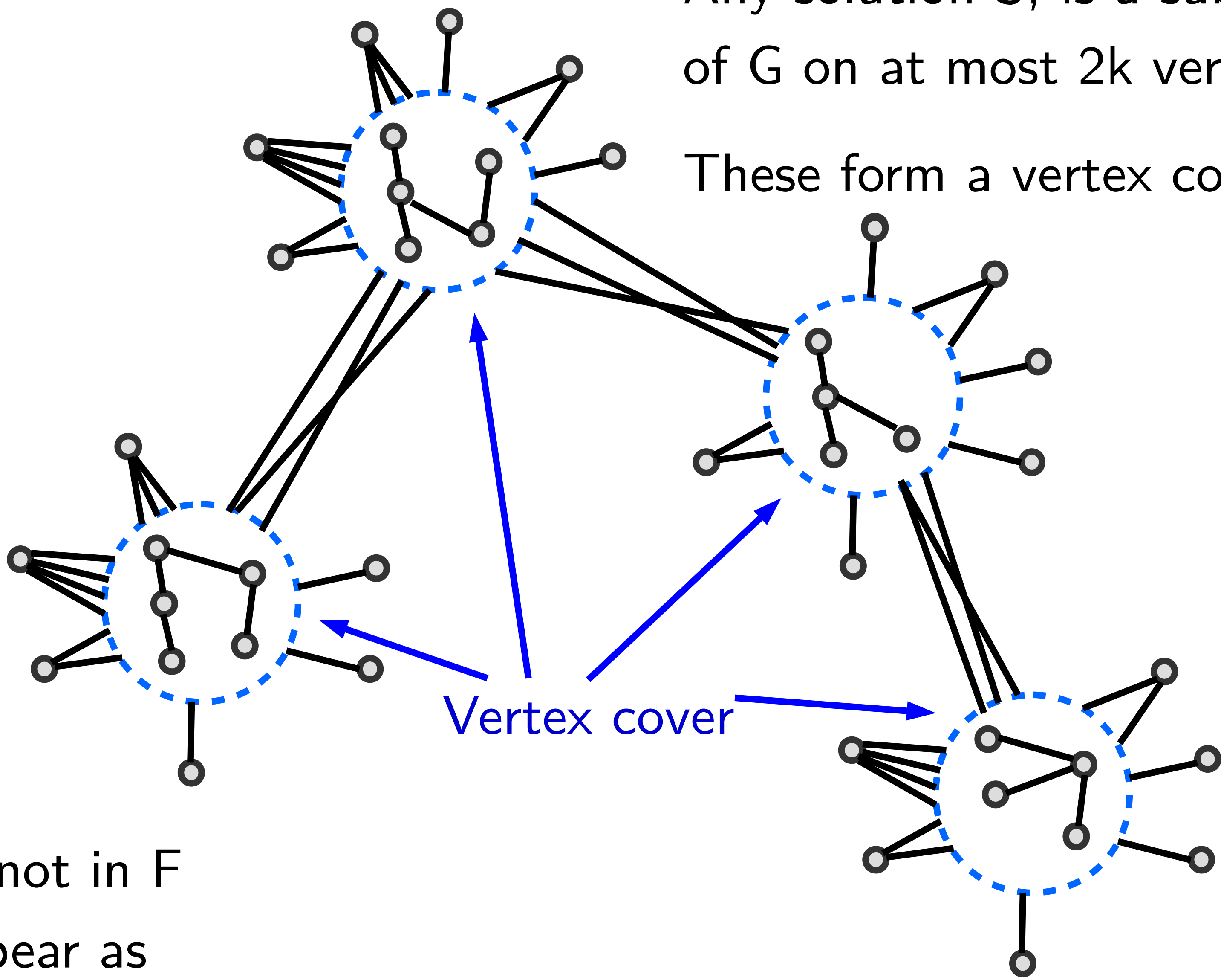


Vertices not in F must appear as "leaves"

Tree Contraction

Any solution S, is a sub-forest of G on at most 2k vertices.

These form a vertex cover



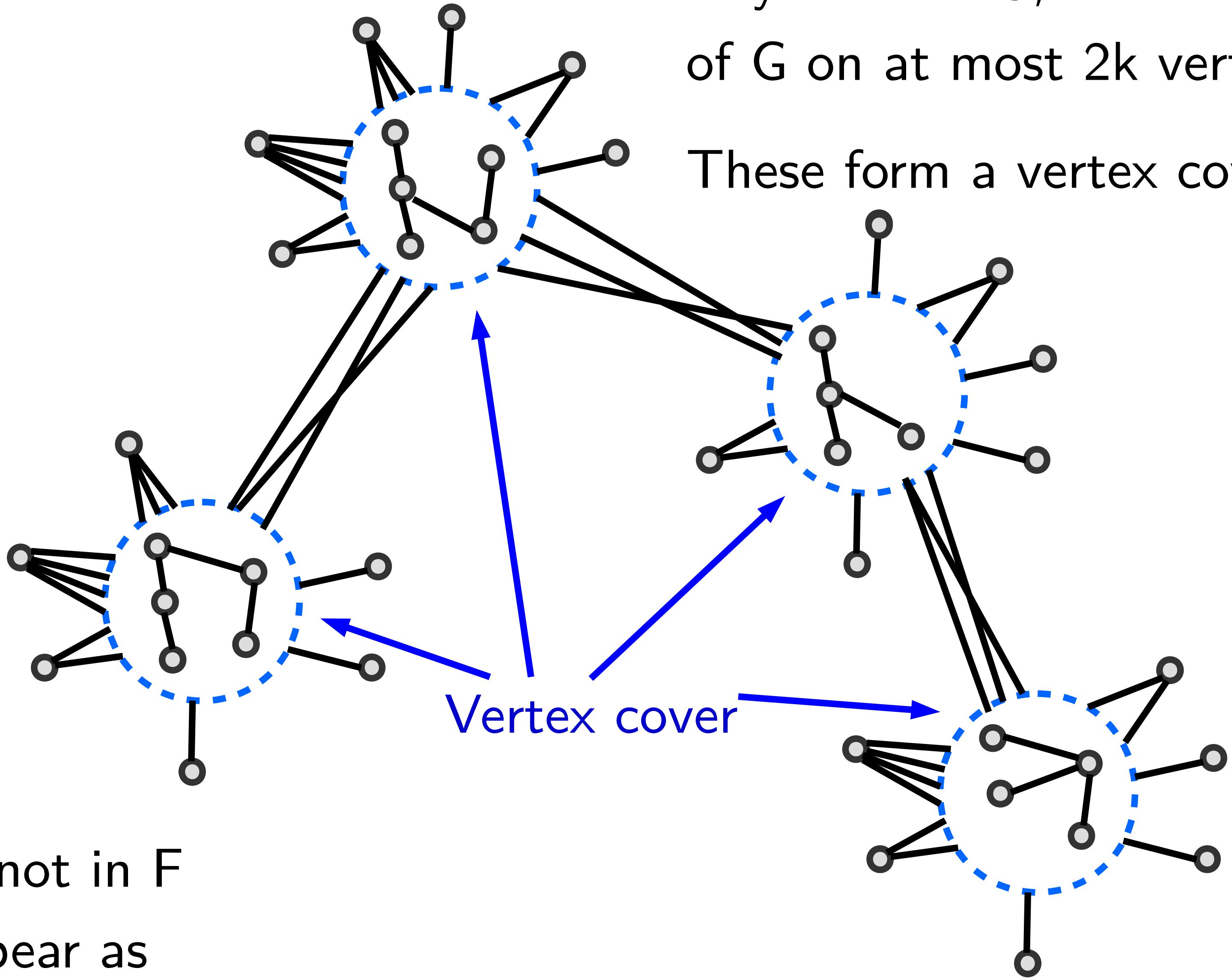
Vertices not in F must appear as "leaves"

If G is k-contractible to a tree, G has CVC of size 2k.

Tree Contraction

Any solution S, is a sub-forest of G on at most 2k vertices.

These form a vertex cover

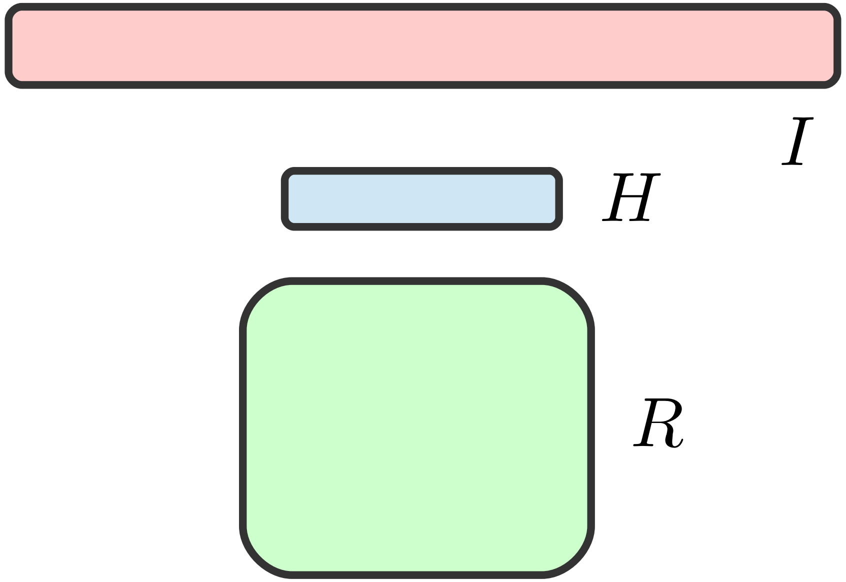


Vertices not in F must appear as "leaves"

Our goal is to reduce the leaves

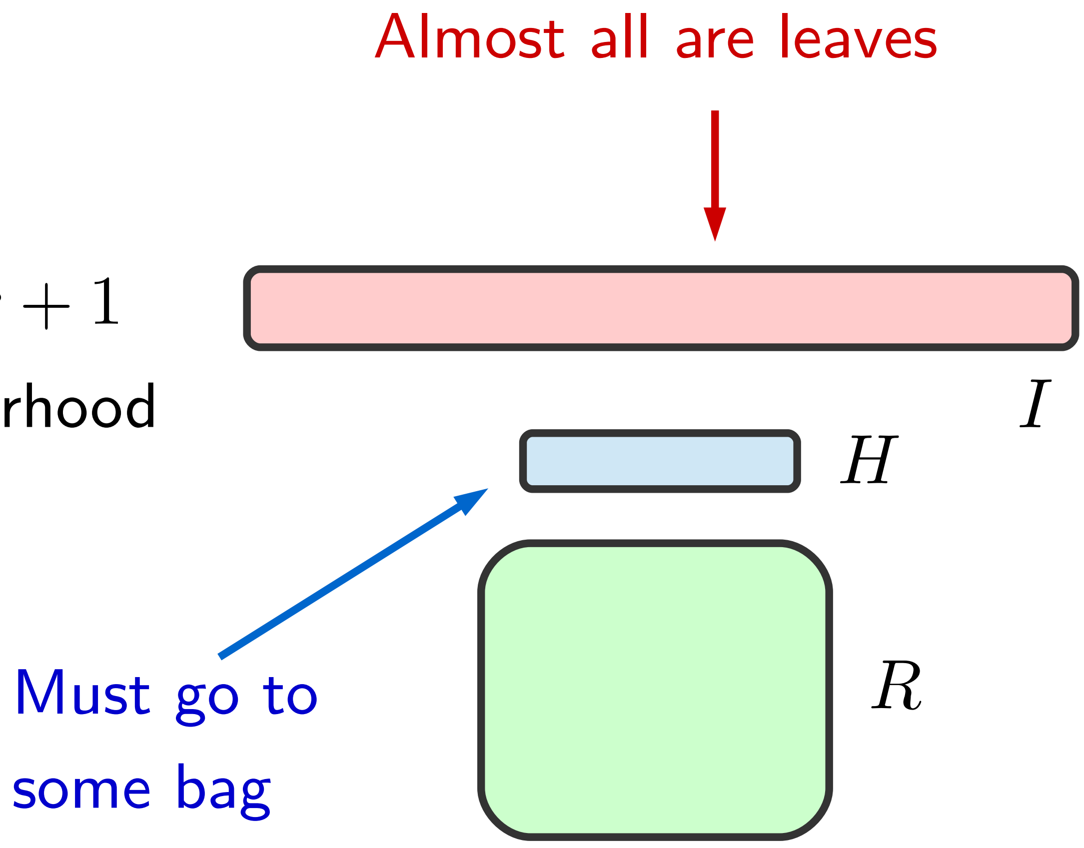
Tree Contraction

- H : vertices of degree $\geq 2k+1$
- I : vertices whose neighborhood is contained in H
- R : the remaining vertices



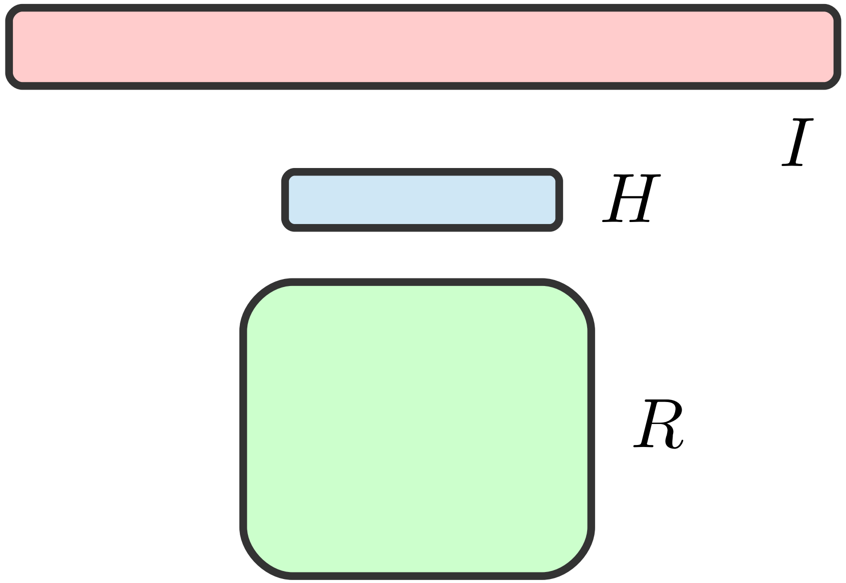
Tree Contraction

- H : vertices of degree $\geq 2k+1$
- I : vertices whose neighborhood is contained in H
- R : the remaining vertices



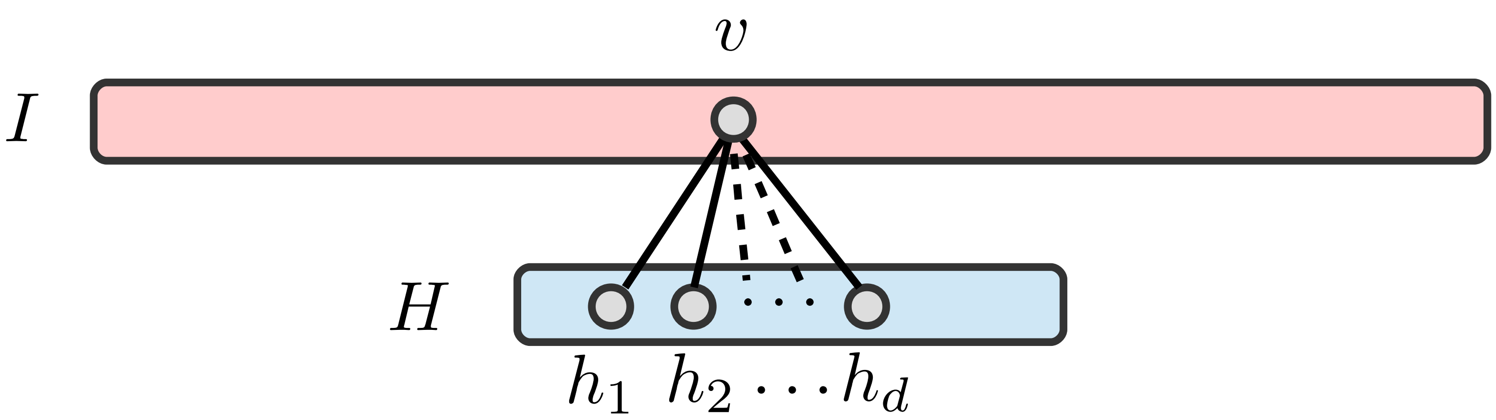
Tree Contraction

- H : vertices of degree $\geq 2k+1$
- I : vertices whose neighborhood is contained in H
- R : the remaining vertices



Goal is to reduce I

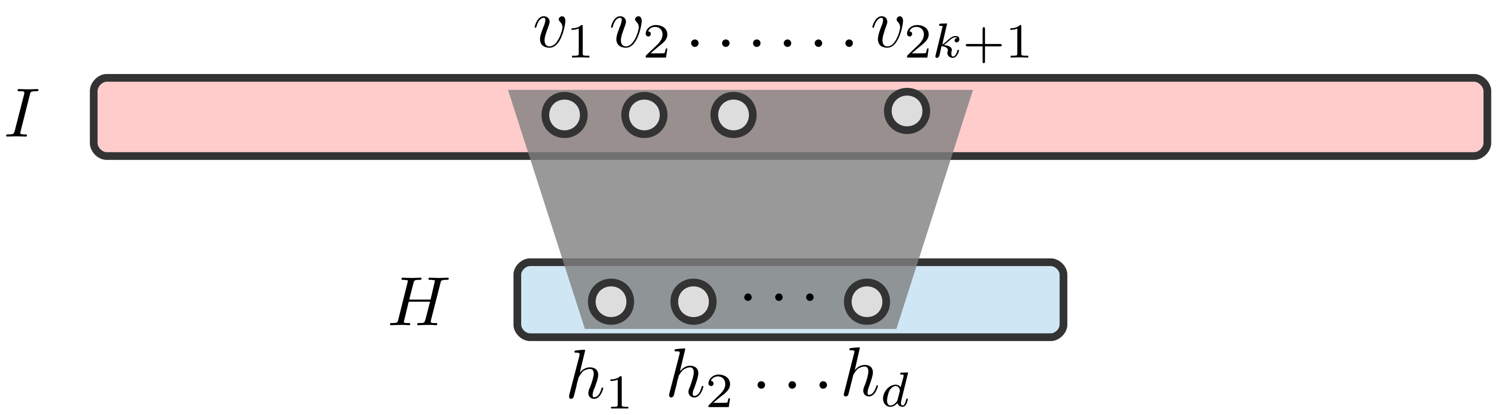
Tree Contraction



We don't know if $h_1, h_2, \dots, h_d$ are in the same bag

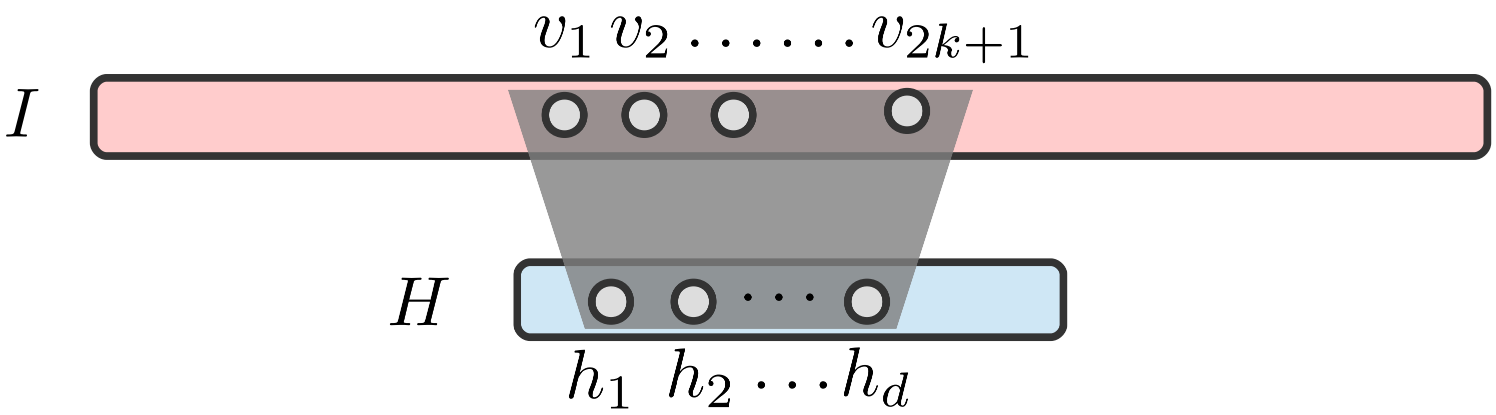
$$d = \lceil \frac{\alpha}{\alpha - 1} \rceil$$

Tree Contraction



$$d = \lceil \frac{\alpha}{\alpha - 1} \rceil$$

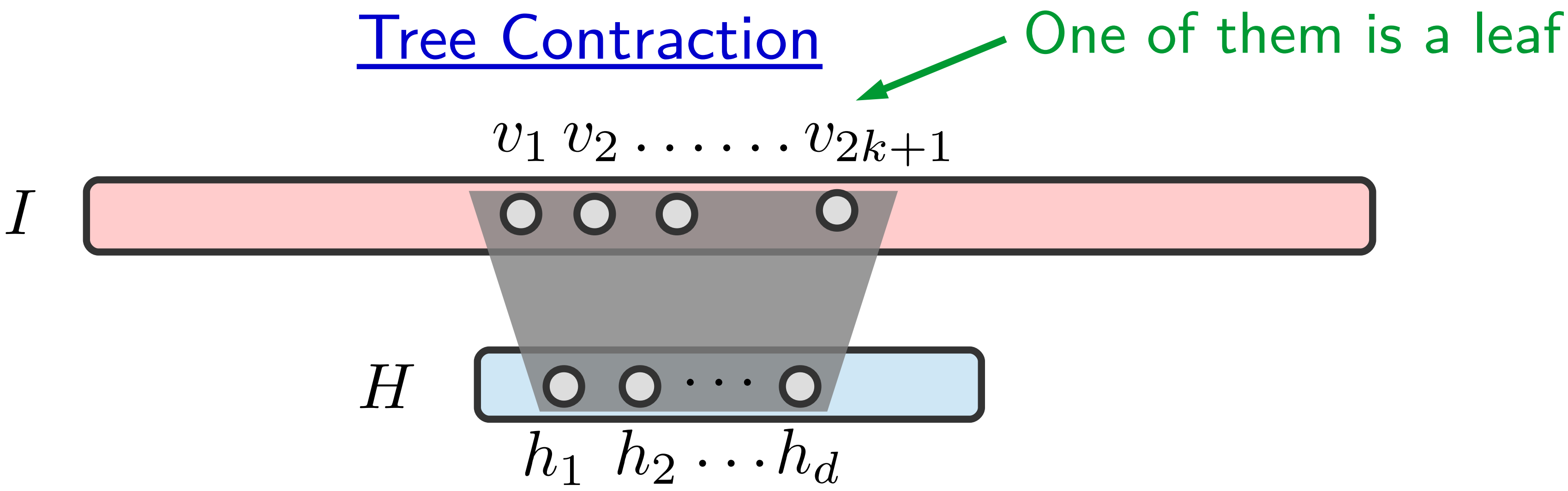
Tree Contraction



Claim: $h_1, h_2, \ldots, h_d$ must all go to the same bag

$$d = \lceil \frac{\alpha}{\alpha - 1} \rceil$$

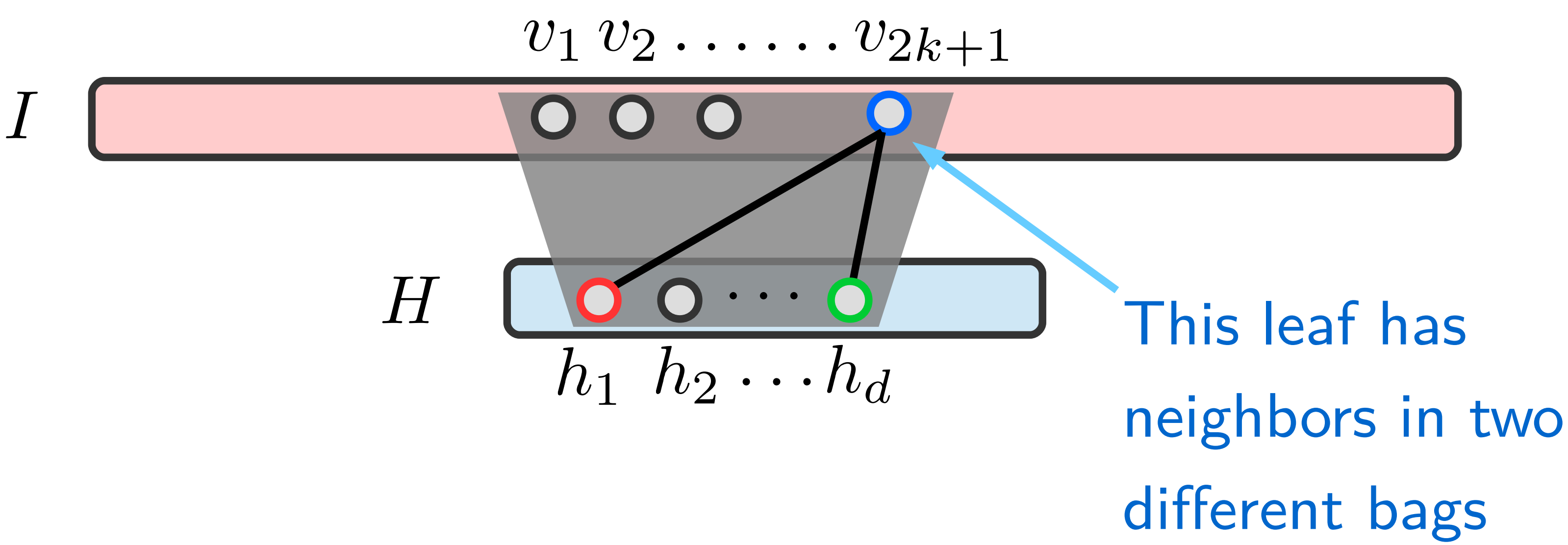
Tree Contraction



Claim: $h_1, h_2, \ldots, h_d$ must all go to the same bag

$$d = \lceil \frac{\alpha}{\alpha - 1} \rceil$$

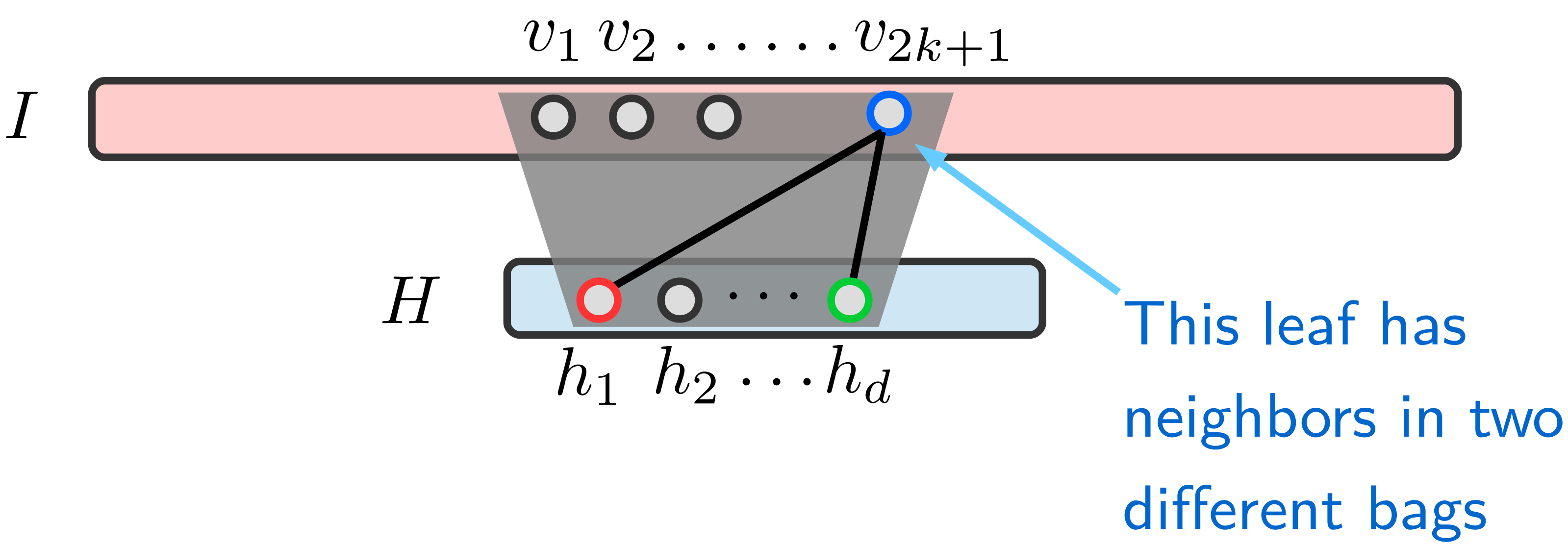
Tree Contraction



Claim: $h_1, h_2, \ldots, h_d$ must all go to the same bag

$$d = \lceil \frac{\alpha}{\alpha - 1} \rceil$$

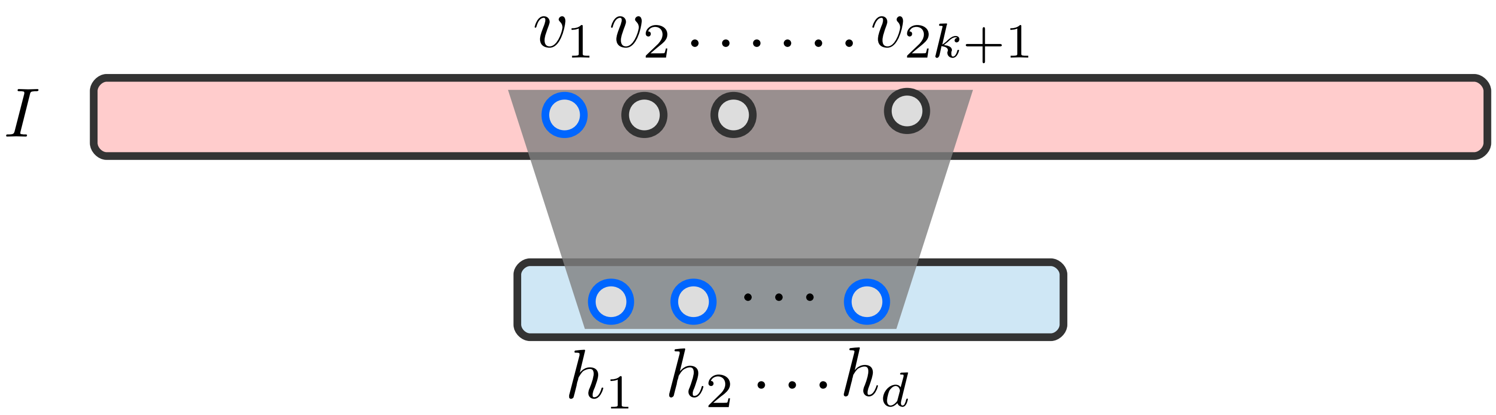
Tree Contraction



Claim: $h_1, h_2, \ldots, h_d$ must all go to the same bag

$$d = \lceil \frac{\alpha}{\alpha - 1} \rceil$$

Tree Contraction



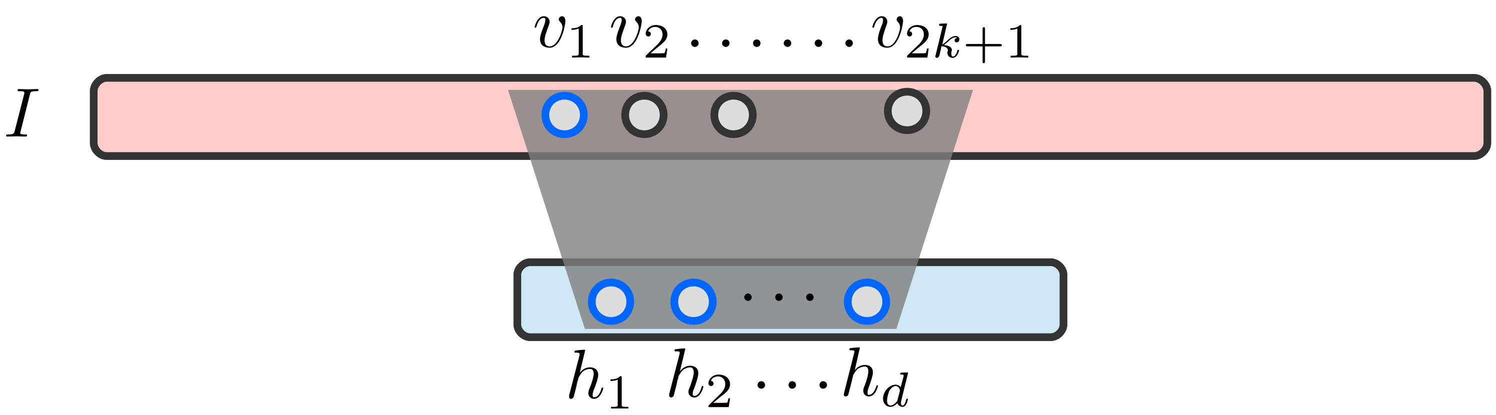
Claim: $h_1, h_2, \dots, h_d$ must all go to the same bag

Reduction Rule :

Contract $\{v_1, h_1, h_2, \dots, h_d\}$ to a vertex in H

$$d = \lceil \frac{\alpha}{\alpha - 1} \rceil$$

Tree Contraction



Claim: $h_1, h_2, \ldots, h_d$ must all go to the same bag

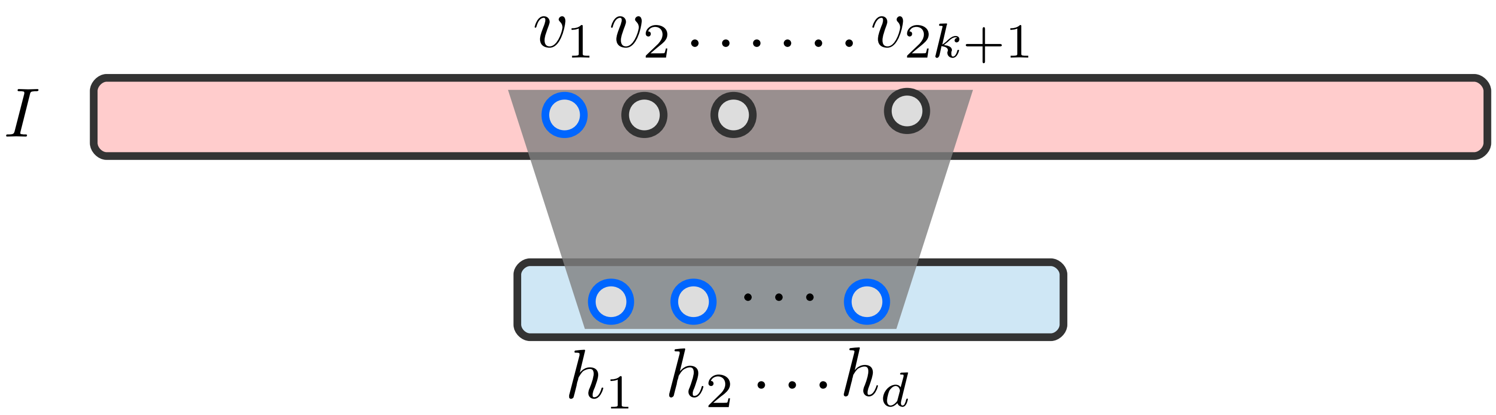
Reduction Rule :

Contract $\{v_1, h_1, h_2, \ldots, h_d\}$ to a vertex in H

This is α-safe

$$d = \lceil \frac{\alpha}{\alpha - 1} \rceil$$

Tree Contraction



Claim: $h_1, h_2, \ldots, h_d$ must all go to the same bag

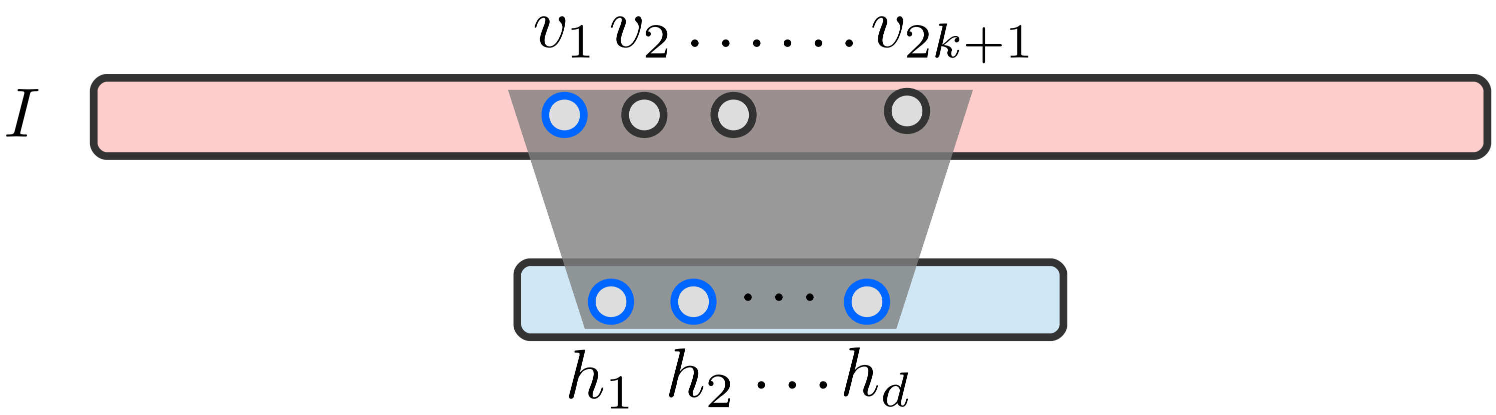
Reduction Rule :

Contract $\{v_1, h_1, h_2, \ldots, h_d\}$ to a vertex in H

We argue that $|I| \leq \mathcal{O}(k^d)$

$$d = \lceil \frac{\alpha}{\alpha - 1} \rceil$$

Tree Contraction



Claim: $h_1, h_2, \ldots, h_d$ must all go to the same bag

Reduction Rule :

Contract $\{v_1, h_1, h_2, \ldots, h_d\}$ to a vertex in H

We get a lossy kernel for Tree Contraction of size $\mathcal{O}(k^d)$

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.

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.

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.