Parameterized Algorithms

Lecture 11: Advanced Kernelization Techniques

July 17, 2020

Max-Planck Institute for Informatics, Germany.

Kernelization.

Compress an instance (X, k) to an instance (X', k') such that $|X'| + |k'| \leq \mathsf{poly}(k)$

We can solve (X, k) in polynomial time given a solution to (X', k')

Kernelization.

Compress an instance (X, k) to an instance (X', k') such that $|X'| + |k'| \leq \mathsf{poly}(k)$

We can solve (X, k) in polynomial time given a solution to (X', k')

Many Problems don't have polynomial kernels

Turing Kernelization

Compress an instance (X, k) to **several** instances $\{(X_i, k_i)\}$ such that $|X_i| + |k_i| \leq \mathsf{poly}(k)$

We can solve (X, k) in polynomial time given solutions to $\{(X_i, k_i)\}$

Lossy Kernelization

Compress an instance (X, k) to an instance (X', k') such that
$|X'| + |k'| \leq \mathsf{poly}(k)$

We can compute an approximate solution to (X, k) from an approximate solution to (X', k')

Turing Kernelization

Max Leaf Subtree:

Given a graph G, and integer k is there a sub-tree with at least k leaves ?

- When G is connected MLS has a polynomial kernel.

Polynomial Kernel for Connected MLS

- **Reduction Rule 1:** Contract a degree 2 vertex with non-adjacent neighbors that are also degree 2.
- **Lemma:** When RR1 is not applicable, and there are more that $6k^2 + k$ vertices, the given instance is a YES instance.

Polynomial Kernel for Connected MLS

- **Reduction Rule 1:** Contract a degree 2 vertex with non-adjacent neighbors that are also degree 2.
- **Lemma:** When RR1 is not applicable, and there are more that $6k^2 + k$ vertices, the given instance is a YES instance.

- Pick a sequence of vertices, S in the following manner
  - Initially all vertices are unmarked.
  - While there is an unmarked vertex of degree ≥ 3:
    - Pick a largest degree unmarked vertex v into S
    - Mark $N^2[v] = \{v\} \cup \{u \mid \mathsf{dist}(u, v) \leq 2\}$
  - Let $S = \{v_1, v_2 \ldots v_r\}$
- **Observation:** $N[v_i] \cap N[v_j] = \emptyset$

Polynomial Kernel for Connected MLS

- **Claim 1:** If $\sum_{i=1}^{r} d(v_i) - 2 \geq k$ then we have a YES instance
- Start with a forest where each v_i is the center of a star, then grow it into a (spanning) tree by connecting these stars with $r-1$ paths.
- The resulting tree has at least $\sum_{i=1}^{r} d(v_i) - 2 \geq k$ leaves.

Polynomial Kernel for Connected MLS

- **Claim 1:** If $\sum_{i=1}^{r} d(v_i) - 2 \geq k$ then we have a YES instance
- Start with a forest where each v_i is the center of a star, then grow it into a (spanning) tree by connecting these stars with $r - 1$ paths.
- The resulting tree has at least $\sum_{i=1}^{r} d(v_i) - 2 \geq k$ leaves.

- **Claim 2:** If $r \geq k$ then we have a YES instance
- Because each $v \in S$ has degree at least 3.

Polynomial Kernel for Connected MLS

- **Claim 3:** If there is a vertex v and a number d such that $|\{u \mid \mathsf{dist}(u,v) = d\}| \geq k$ then we have a YES instance.
- For each vertex u, pick some path of length **exactly** d to v
- The union of these paths is a subtree with $\geq k$ leaves.
- The key observation is that some u_i is not an internal vertex on the path for u_j, as they are both at distance d from v.

Polynomial Kernel for Connected MLS

- **Claim 4:** For some number d, if there at least rk vertices at distance exactly d from S, then we have a YES instance.
- There are r vertices in S, hence there is some vertex in $v \in S$ for which there are at least k vertices at distance exactly d from v
- The previous claim implies we have a YES instance.

Polynomial Kernel for Connected MLS

- Let $N^2[S] = S \cup \{u \mid \mathsf{dist}(v,u) \leq 2 \text{ for some } v \in S\}$.
- **Claim 5:** The number of connected components in $G - N^2[S]$ is at most k^2.
- As G is connected, each connected component of $G - N^2[S]$ has a vertex of distance exactly 3 from S.
- By the above claim, the number of vertices at distance exactly 3 is at most $rk \leq k^2$.

Polynomial Kernel for Connected MLS

- **Claim 6:** Any connected component $G - N^2[S]$ contains at most 4 vertices.
- $H = G - N^2[S]$ contains only vertices of degree 2 or less. So it is a collection of paths, cycles and isolated vertices.
- If some component C of H had 5 vertices, then there will be a degree-2 vertex with two degree-2 neighbors (in G) and RR1 applies

Polynomial Kernel for Connected MLS

- **Claim 7:** If RR1 is not applicable, and we have more than $6k^2 + k$ vertices, then we have a YES instance.

- By Claim 2, $|S| = r \leq k$ (else a YES instance)
- By Claim 3, for $d = 1, 2$ there are at most $2k^2$ vertices at distance 1 or 2 from S, (else a YES instance)
- Hence $|N^2[S]| = k + 2k^2$.
- By Claim 5, number of connected components in $G - N^2[S]$ is at most k^2, (else a YES instance)
- By Claim 6, each connected component has at most 4 vertices. Hence total number of vertices in $G - N^2[S]$ is at most $4k^2$, (else RR1 is applicable)
- In total there can be at most $6k^2 + k$ vertices. Otherwise, we already have a YES instance, or RR1 is applicable.

Max Leaf Subtree:

Given a graph G, and integer k is there a sub-tree with at least k leaves ?

- When G is connected MLS has a polynomial kernel.

Max Leaf Subtree:

Given a graph G, and integer k is there a sub-tree with at least k leaves ?

- When G is connected MLS has a polynomial kernel.
- However, when G is disconnected MLS has no polynomial kernel.

Max Leaf Subtree:

Given a graph G, and integer k is there a sub-tree with at least k leaves ?

- When G is connected MLS has a polynomial kernel.
- However, when G is disconnected MLS has no polynomial kernel.

OR Composition: Take a disjoint union of connected MLS instances

Turing Kernelization

Definition (Turing Kernel)

Let Q be a parameterized problem, and let $f : \mathbb{N} \to \mathbb{N}$ be a computable function. A **Turing Kernel** for Q of size f is an algorithm that can decide if an instance of the problem is a YES instance in polynomial time, given access to an Oracle that solves instance of size $f(k)$ in unit time.

- Max Leaf Subtree admits a Turing Kernel of size $6k^2 + k$.
- Just kernelize each connected component separately
- Then if, any component (and it's kernel) is a YES instance, then the input is a YES instance.

Turing Kernelization

- For MLS, we produced $O(n)$ Turing Kernels, all independent of each other, more or less directly.
- However, we can also produce Turing Kernels in a more complex ways.

  the i-th kernel depends on the Oracle's answers to the previous $(i-1)$ kernels

  .
- Such kind of Turing Kernels are known for k-Path on certain graph classes.
- There is some lower-bound machinery, such as Steiner Tree and Connected Vertex Cover are unlikely to admit Turing Kernels.

Lossy Kernels

Kernelization + Approximation

Kernelization

- Formal Study of Preprocessing / Data Reduction Heuristics

Kernelization

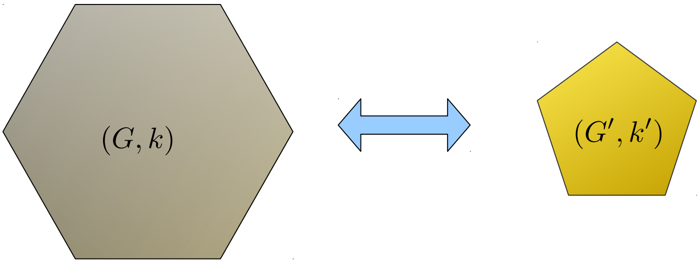
- A parameterized language is defined as $L \subseteq \Sigma^* \times \mathbb{N}$ where, Σ is a finite alphabet.

- A parameterized problem w.r.t L is to decide if a given $(x, k) \in \Sigma^* \times \mathbb{N}$ is in the language or not.
  - (x, k) is called a parameterized instance.

$L_{VC} = \{(G, k) \mid G$ has a Vertex Cover of size $k\}$

Kernelization

- Formal Study of Preprocessing / Data Reduction Heuristics



Given an instance (G, k), run a polynomial time algorithm and produce an instance (G', k') such that,

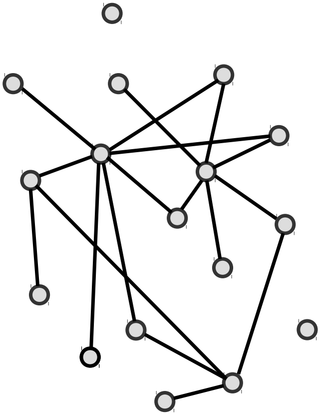
- both instances are **Equivalent**
- $|G'|, |k'| \leq \text{poly}(k)$

The polynomial time algorithm is called a Kernelization Algorithm

A kernel for Vertex Cover

Input: A graph G *and a number* k

Output: A graph G' with at most $2k^2$ vertices and a number $k' \leq k$



$k = 6$

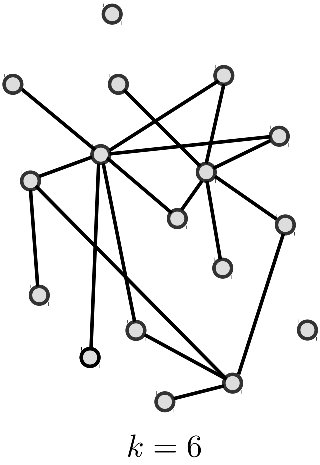
A kernel for Vertex Cover

Input: A graph G *and a number* k

Output: A graph G' with at most $2k^2$ vertices and a number $k' \leq k$

Reduction Rule 1 :

Remove all vertices of degree 0



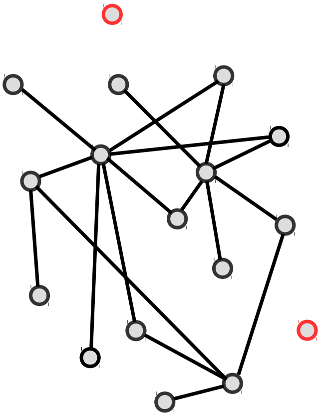
A kernel for Vertex Cover

Input: A graph G *and a number* k

Output: A graph G' with at most $2k^2$ vertices and a number $k' \leq k$

Reduction Rule 1 :

Remove all vertices of degree 0



$k = 6$

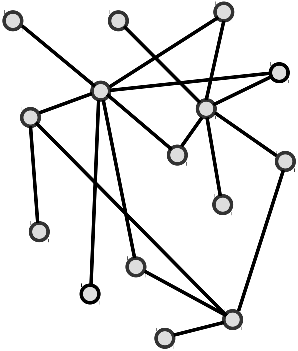
A kernel for Vertex Cover

Input: A graph G *and a number* k

Output: A graph G' with at most $2k^2$ vertices and a number $k' \leq k$

Reduction Rule 1 :

Remove all vertices of degree 0



$k = 6$

A kernel for Vertex Cover

Input: A graph G *and a number* k

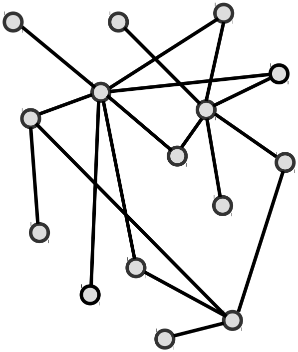
Output: A graph G' with at most $2k^2$ vertices and a number $k' \leq k$

Reduction Rule 1 :

Remove all vertices of degree 0

Reduction Rule 2 :

Remove a vertex of degree $\geq k+1$ and decrease k by 1



$k = 6$

A kernel for Vertex Cover

Input: A graph G *and a number* k

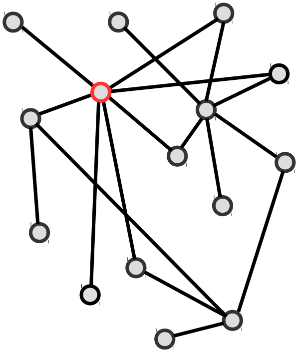
Output: A graph G' with at most $2k^2$ vertices and a number $k' \leq k$

Reduction Rule 1 :

Remove all vertices of degree 0

Reduction Rule 2 :

Remove a vertex of degree $\geq k+1$ and decrease k by 1



$k = 6$

A kernel for Vertex Cover

Input: A graph G *and a number* k

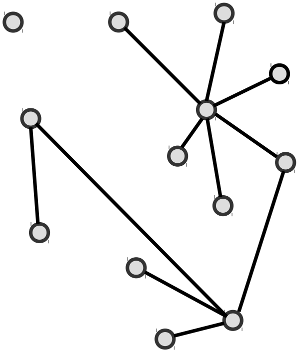
Output: A graph G' with at most $2k^2$ vertices and a number $k' \leq k$

Reduction Rule 1 :

Remove all vertices of degree 0

Reduction Rule 2 :

Remove a vertex of degree $\geq k+1$ and decrease k by 1



$k = 5$

A kernel for Vertex Cover

Input: A graph G *and a number* k

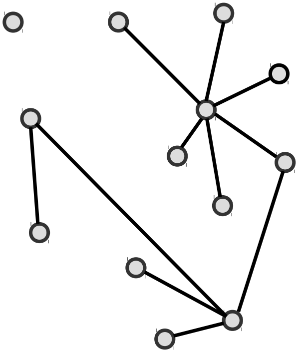
Output: A graph G' with at most $2k^2$ vertices and a number $k' \leq k$

Reduction Rule 1 :

Remove all vertices of degree 0

Reduction Rule 2 :

Remove a vertex of degree $\geq k+1$ and decrease k by 1



$k = 5$

Apply these Reduction Rules Exhaustively !

A kernel for Vertex Cover

Input: A graph G *and a number* k

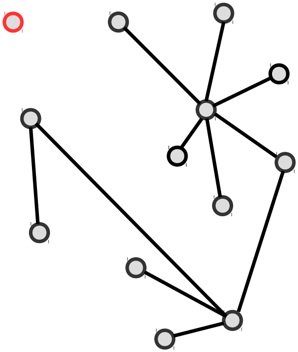
Output: A graph G' with at most $2k^2$ vertices and a number $k' \leq k$

Reduction Rule 1 :

Remove all vertices of degree 0

Reduction Rule 2 :

Remove a vertex of degree $\geq k+1$ and decrease k by 1



$k = 5$

Apply these Reduction Rules Exhaustively !

A kernel for Vertex Cover

Input: A graph G *and a number* k

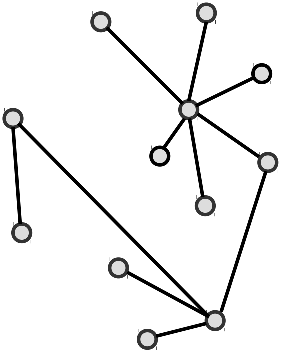
Output: A graph G' with at most $2k^2$ vertices and a number $k' \leq k$

Reduction Rule 1 :

Remove all vertices of degree 0

Reduction Rule 2 :

Remove a vertex of degree $\geq k+1$ and decrease k by 1



$k = 5$

Apply these Reduction Rules Exhaustively !

A kernel for Vertex Cover

Input: A graph G *and a number* k

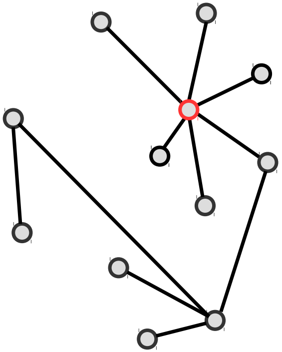
Output: A graph G' with at most $2k^2$ vertices and a number $k' \leq k$

Reduction Rule 1 :

Remove all vertices of degree 0

Reduction Rule 2 :

Remove a vertex of degree $\geq k+1$ and decrease k by 1



Apply these Reduction Rules Exhaustively !

A kernel for Vertex Cover

Input: A graph G *and a number* k

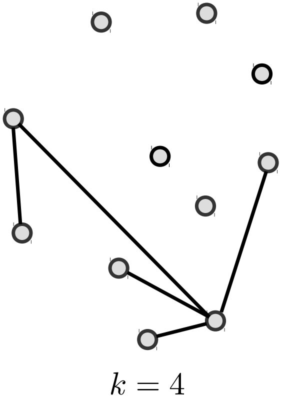
Output: A graph G' with at most $2k^2$ vertices and a number $k' \leq k$

Reduction Rule 1 :

Remove all vertices of degree 0

Reduction Rule 2 :

Remove a vertex of degree $\geq k+1$ and decrease k by 1



Apply these Reduction Rules Exhaustively !

A kernel for Vertex Cover

Input: A graph G *and a number* k

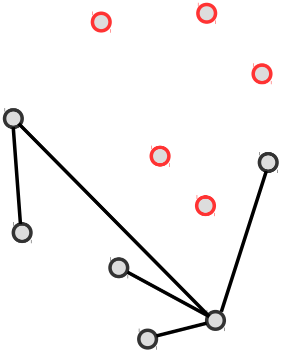
Output: A graph G' with at most $2k^2$ vertices and a number $k' \leq k$

Reduction Rule 1 :

Remove all vertices of degree 0

Reduction Rule 2 :

Remove a vertex of degree $\geq k+1$ and decrease k by 1



Apply these Reduction Rules Exhaustively !

A kernel for Vertex Cover

Input: A graph G *and a number* k

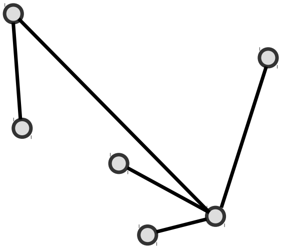
Output: A graph G' with at most $2k^2$ vertices and a number $k' \leq k$

Reduction Rule 1 :

Remove all vertices of degree 0

Reduction Rule 2 :

Remove a vertex of degree $\geq k+1$ and decrease k by 1



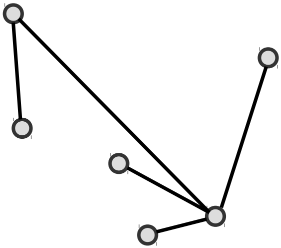
$k = 4$

Apply these Reduction Rules Exhaustively !

A kernel for Vertex Cover

Input: A graph G *and a number* k

Output: A graph G' with at most $2k^2$ vertices and a number $k' \leq k$



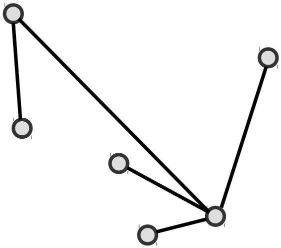
$k = 4$

A kernel for Vertex Cover

Input: A graph G *and a number* k

Output: A graph G' with at most $2k^2$ vertices and a number $k' \leq k$

Observation 1 : Every vertex has at least one edge incident on it



$k = 4$

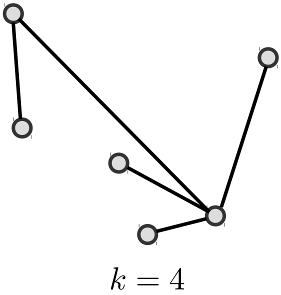
A kernel for Vertex Cover

Input: A graph G *and a number* k

Output: A graph G' with at most $2k^2$ vertices and a number $k' \leq k$

Observation 1 : Every vertex has at least one edge incident on it

Observation 2 : Every vertex has at most k edges incident on it

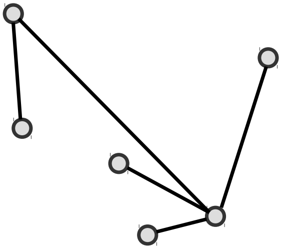


A kernel for Vertex Cover

Input: A graph G *and a number* k

Output: A graph G' with at most $2k^2$ vertices and a number $k' \leq k$

Observation 3: Either G has at most k^2 edges (and at most $2k^2$ vertices)



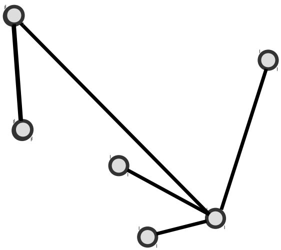
$k = 4$

A kernel for Vertex Cover

Input: A graph G *and a number* k

Output: A graph G' with at most $2k^2$ vertices and a number $k' \leq k$

Observation 3: Either G has at most k^2 edges (and at most $2k^2$ vertices)



$k = 4$

Output : $G' \leftarrow G$ and $k' \leftarrow k$

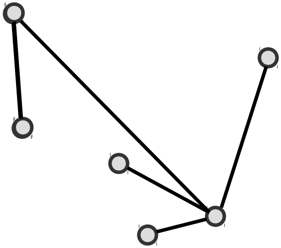
A kernel for Vertex Cover

Input: A graph G *and a number* k

Output: A graph G' with at most $2k^2$ vertices and a number $k' \leq k$

Observation 3: Either G has at most k^2 edges (and at most $2k^2$ vertices)

Or there are more than k^2 edges, which cannot be covered by k vertices of degree k



$k = 4$

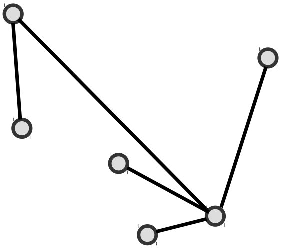
A kernel for Vertex Cover

Input: A graph G *and a number* k

Output: A graph G' with at most $2k^2$ vertices and a number $k' \leq k$

Observation 3: Either G has at most k^2 edges (and at most $2k^2$ vertices)

Or there are more than k^2 edges, which cannot be covered by k vertices of degree k

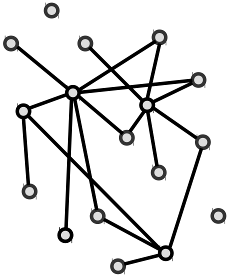


$k = 4$

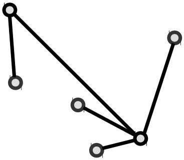
Output : $G' \leftarrow$ any $k^2 + 1$ edges of G and $k' \leftarrow k$

A kernel for Vertex Cover

- Thus (G,k) and (G',k') are equivalent instances and $|G'|, |k'| \leq 2k^2$



$k = 6$

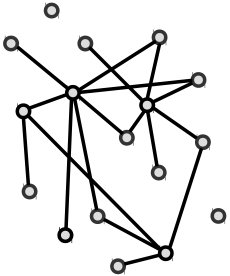


$k' = 4$

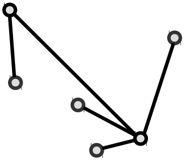
A kernel for Vertex Cover

- Thus (G, k) and (G', k') are equivalent instances and $|G'|, |k'| \leq 2k^2$

Can we compute a solution to (G, k) if we are given a solution to (G', k') ?



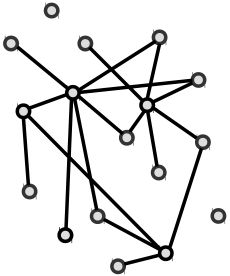
$k = 6$



$k' = 4$

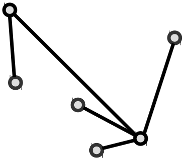
A kernel for Vertex Cover

- Thus (G, k) and (G', k') are equivalent instances and $|G'|, |k'| \leq 2k^2$



$k = 6$

Can we compute a solution to (G, k) if we are given a solution to (G', k') ?

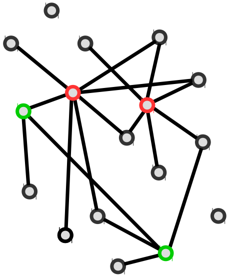


$k' = 4$

Yes we can !

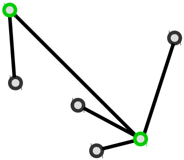
A kernel for Vertex Cover

- Thus (G, k) and (G', k') are equivalent instances and $|G'|, |k'| \leq 2k^2$



$k = 6$

Can we compute a solution to (G, k) if we are given a solution to (G', k') ?



$k' = 4$

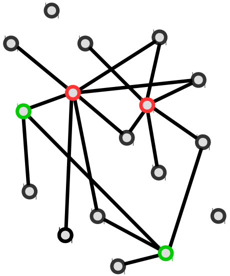
Yes we can !

A kernel for Vertex Cover

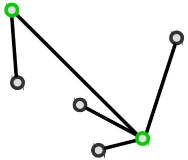
- Thus (G, k) and (G', k') are equivalent instances and $|G'|, |k'| \leq 2k^2$

Can we compute a solution to (G, k) if we are given a solution to (G', k') ?

This is true of many kernelization algorithms for many problems.



$k = 6$

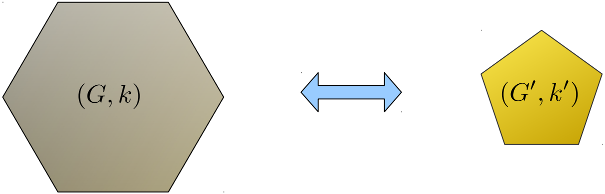


$k' = 4$

Kernels and Optimization Problems

Given an instance (G, k), run a polynomial time algorithm and produce an instance (G', k') such that,

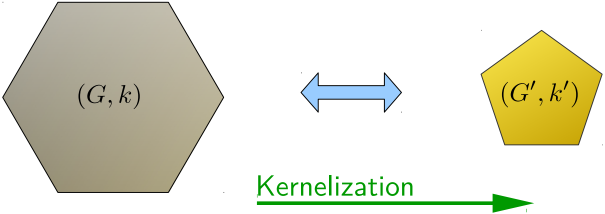
- both instances are Equivalent
- $|G'|, |k'| \leq \mathsf{poly}(k)$



Kernels and Optimization Problems

Given an instance (G, k), run a polynomial time algorithm and produce an instance (G', k') such that,

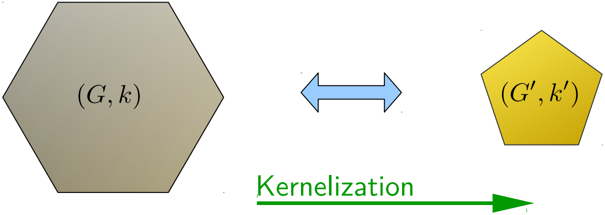
- both instances are Equivalent
- $|G'|, |k'| \leq \text{poly}(k)$



Kernels and Optimization Problems

Given an instance (G, k), run a polynomial time algorithm and produce an instance (G', k') such that,

- both instances are Equivalent
- $|G'|, |k'| \leq \text{poly}(k)$

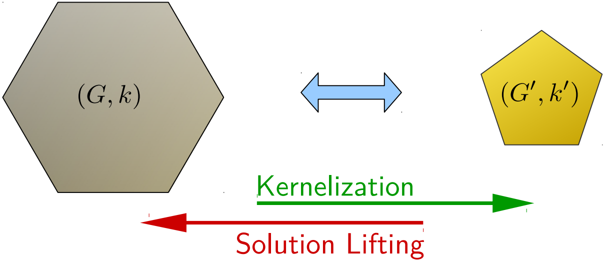


Take a solution S' of (G', k') and turn it into a solution S for (G, k)

Kernels and Optimization Problems

Given an instance (G,k), run a polynomial time algorithm and produce an instance (G',k') such that,

- both instances are Equivalent
- $|G'|, |k'| \leq \text{poly}(k)$

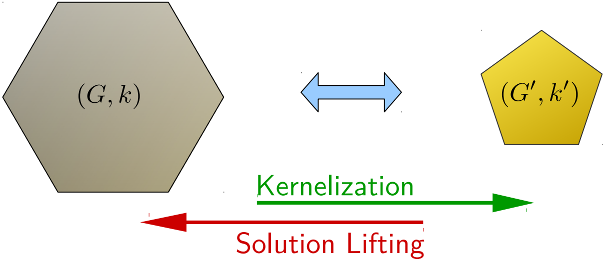


Take a solution S' of (G',k') and turn it into a solution S for (G,k)

Kernels and Optimization Problems

Given an instance (G, k), run a polynomial time algorithm and produce an instance (G', k') such that,

- both instances are Equivalent
- $|G'|, |k'| \leq \text{poly}(k)$



Take a solution S' of (G', k') and turn it into a solution S for (G, k)

But what is the "**quality**" of the solution S compared to S' ?

Kernels and Optimization Problems

Given an instance (G, k), run a polynomial time algorithm and produce an instance (G', k') such that,

- both instances are Equivalent
- $|G'|, |k'| \leq \text{poly}(k)$

We need some more definitions :)

Kernels and Optimization Problems

Given an instance (G, k), run a polynomial time algorithm and produce an instance (G', k') such that,

- both instances are Equivalent
- $|G'|, |k'| \leq \text{poly}(k)$

A parameterized minimization problem is defined as

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

Kernels and Optimization Problems

Given an instance (G, k), run a polynomial time algorithm and produce an instance (G', k') such that,

- both instances are Equivalent
- $|G'|, |k'| \leq \text{poly}(k)$

A parameterized minimization problem is defined as

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

Graph Parameter Solution set

Kernels and Optimization Problems

Given an instance (G, k), run a polynomial time algorithm and produce an instance (G', k') such that,

- both instances are Equivalent
- $|G'|, |k'| \leq \text{poly}(k)$

A parameterized minimization problem is defined as

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

Graph Parameter Solution set

$$\Pi_{VC}(G, k, S) = \begin{cases} \infty & \text{if } S \text{ is not a vertex cover} \\ \min\{|S|, k+1\} & \text{otherwise} \end{cases}$$

Kernels and Optimization Problems

Given an instance (G, k), run a polynomial time algorithm and produce an instance (G', k') such that,

- both instances are Equivalent
- $|G'|, |k'| \leq \text{poly}(k)$

A parameterized minimization problem is defined as

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

Graph Parameter Solution set

$$\Pi_{VC}(G, k, S) = \begin{cases} \infty & \text{if } S \text{ is not a vertex cover} \\ \min\{|S|, k+1\} & \text{otherwise} \end{cases}$$

Value of the solution S

Kernels and Optimization Problems

Given an instance (G, k), run a polynomial time algorithm and produce an instance (G', k') such that,

- both instances are Equivalent
- $|G'|, |k'| \leq \text{poly}(k)$

A parameterized minimization problem is defined as

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

Graph Parameter Solution set

$$\Pi_{VC}(G, k, S) = \begin{cases} \infty & \text{if } S \text{ is not a vertex cover} \\ \min\{|S|, k+1\} & \text{otherwise} \end{cases}$$

Value of the solution S

We are only interested in solutions of cardinality $\leq k$

Kernels and Optimization Problems

Given an instance (G, k), run a polynomial time algorithm and produce an instance (G', k') such that,

- both instances are Equivalent
- $|G'|, |k'| \leq \text{poly}(k)$

A parameterized minimization problem is defined as

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

$$\Pi_{VC}(G, k, S) = \begin{cases} \infty & \text{if } S \text{ is not a vertex cover} \\ \min\{|S|, k+1\} & \text{otherwise} \end{cases}$$

Kernels and Optimization Problems

Given an instance (G, k), run a polynomial time algorithm and produce an instance (G', k') such that,

- both instances are Equivalent
- $|G'|, |k'| \leq \text{poly}(k)$

A parameterized minimization problem is defined as

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

$$\Pi_{VC}(G, k, S) = \begin{cases} \infty & \text{if } S \text{ is not a vertex cover} \\ \min\{|S|, k+1\} & \text{otherwise} \end{cases}$$

If S^* is an optimum solution to (G, k), then for any S the quality of S is $\dfrac{\Pi(G, k, S)}{\Pi(G, k, S^*)}$

Kernels and Optimization Problems

Given an instance (G, k), run a polynomial time algorithm and produce an instance (G', k') such that,

- both instances are Equivalent
- $|G'|, |k'| \leq \text{poly}(k)$

A parameterized minimization problem is defined as

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

$$\Pi_{VC}(G, k, S) = \begin{cases} \infty & \text{if } S \text{ is not a vertex cover} \\ \min\{|S|, k+1\} & \text{otherwise} \end{cases}$$

If S^* is an optimum solution to (G, k), then for any S the quality of S is $\dfrac{\Pi(G, k, S)}{\Pi(G, k, S^*)}$ a.k.a the Approximation Ratio

Kernels and Optimization Problems

Given an instance (G, k), run a polynomial time algorithm and produce an instance (G', k') such that,

- both instances are Equivalent
- $|G'|, |k'| \leq \text{poly}(k)$

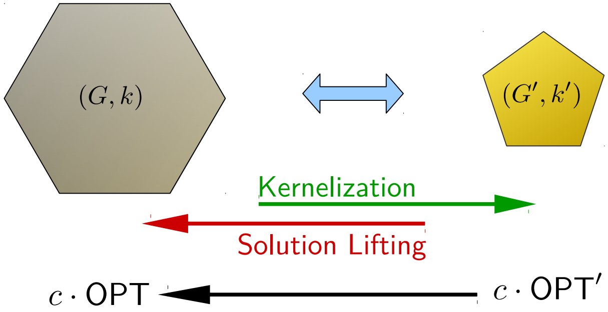
A parameterized minimization problem is defined as

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

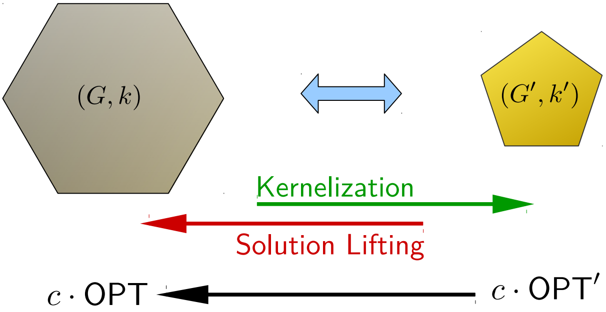
Given a quality c solution to (G', k') find a solution to (G, k) of the same quality in polynomial time !

If S^* is an optimum solution to (G, k), then for any S the quality of S is $\dfrac{\Pi(G,k,S)}{\Pi(G,k,S^*)}$ a.k.a the Approximation Ratio

Kernels and Optimization Problems

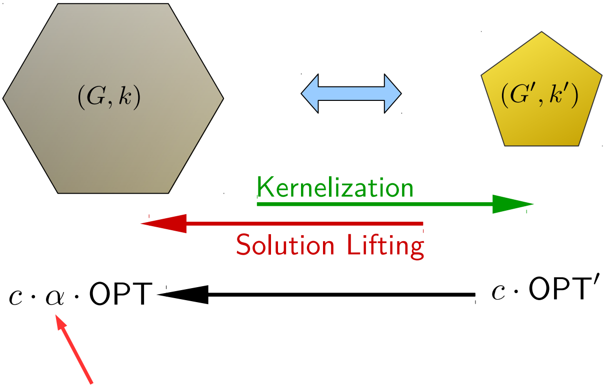


Kernels and Optimization Problems



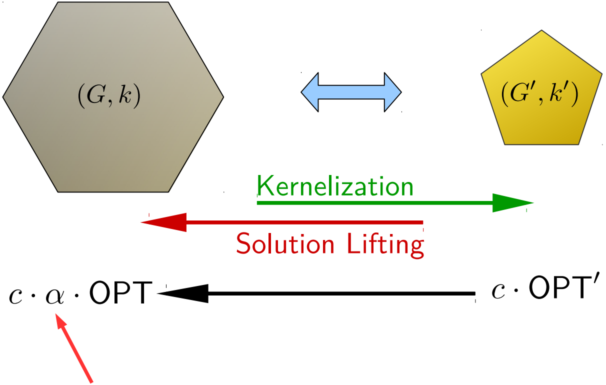
Lets generalize this notion a bit more

Kernels and Optimization Problems



Allow a loss factor in kernelization / solution lifting process

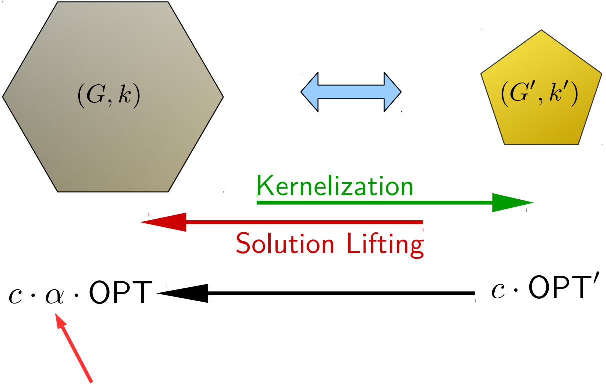
Kernels and Optimization Problems



Allow a loss factor in kernelization / solution lifting process

Lossy Kernels !

Kernels and Optimization Problems



Allow a loss factor in kernelization / solution lifting process

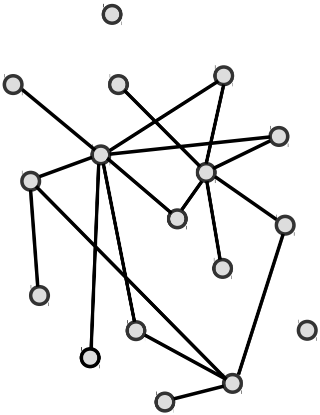
Lossy Kernels !

But why do we need this notion ?

Connected Vertex Cover

Input: A graph G *and a number* k

Question: Is there a vertex cover of value k that is also connected ?



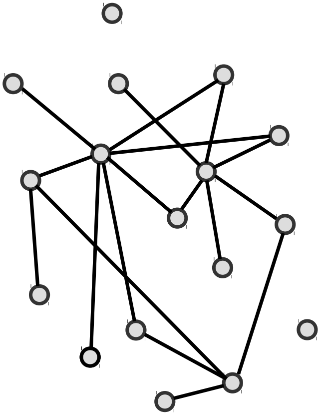
$k = 6$

Connected Vertex Cover

Input: A graph G *and a number* k

Question: Is there a vertex cover of value k that is also connected ?

This problem cannot have a Polynomial Kernel.



$k = 6$

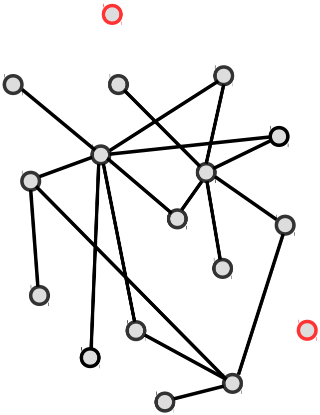
Connected Vertex Cover

Input: A graph G *and a number* k

Question: Is there a vertex cover of value k that is also connected ?

Reduction Rule 1 :

Remove all vertices of degree 0



$k = 6$

Connected Vertex Cover

Input: A graph G *and a number* k

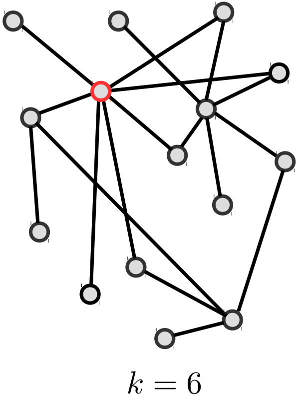
Question: Is there a vertex cover of value k that is also connected ?

Reduction Rule 1 :

Remove all vertices of degree 0

Reduction Rule 2 :

Remove a vertex of degree $\geq k+1$ and decrease k by 1



Connected Vertex Cover

Input: A graph G *and a number* k

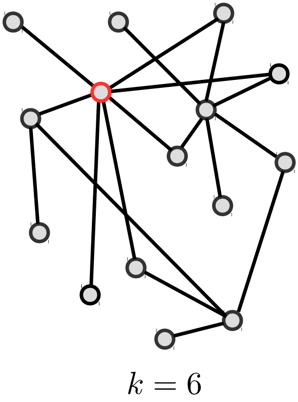
Question: Is there a vertex cover of value k that is also connected ?

Reduction Rule 1 :

Remove all vertices of degree 0

Reduction Rule 2 :

Remove a vertex of degree $\geq k+1$ and decrease k by 1



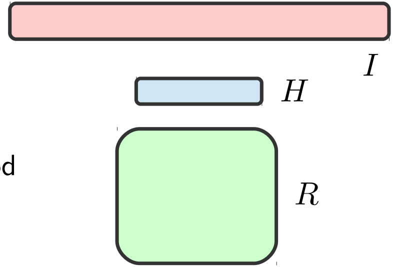
We lose information about connectivity !

Connected Vertex Cover

Input: A graph G *and a number* k

Question: Is there a vertex cover of value k that is also connected ?

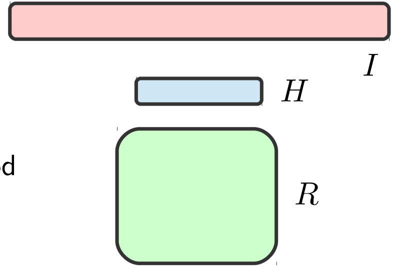
- H : vertices of degree $\geq k+1$
- I : vertices whose neighborhood is contained in H
- R : the remaining vertices



Connected Vertex Cover

Input: A graph G *and a number* k

Question: Is there a vertex cover of value k that is also connected ?



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

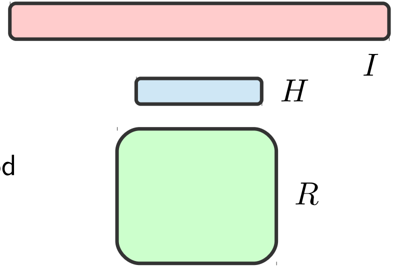
Normally, we remove I but they connect subsets of H

And $|I|$ could be very large.

Connected Vertex Cover

Input: A graph G *and a number* k

Question: Is there a vertex cover of value k that is also connected ?



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

Normally, we remove I but they connect subsets of H

And $|I|$ could be very large.

But this problem admits a lossy polynomial kernel !

Safe Reduction Rules

- A kernelization algorithm sequence of applications reduction rules.

Safe Reduction Rules

- A kernelization algorithm sequence of applications reduction rules.

$$(I, k) \iff (I_1, k_1) \iff (I_2, k_2) \iff \ldots\ldots \iff (I_\ell, k_\ell)$$

Safe Reduction Rules

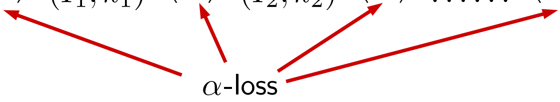
- A kernelization algorithm sequence of applications reduction rules.
- Applying a α-lossy reduction rule ℓ times leads to a α^ℓ loss.

$$(I, k) \iff (I_1, k_1) \iff (I_2, k_2) \iff \ldots\ldots \iff (I_\ell, k_\ell)$$

Safe Reduction Rules

- A kernelization algorithm sequence of applications reduction rules.
- Applying a α-lossy reduction rule ℓ times leads to a α^{ℓ} loss.

$$(I, k) \iff (I_1, k_1) \iff (I_2, k_2) \iff \ldots\ldots \iff (I_\ell, k_\ell)$$



Safe Reduction Rules

- A kernelization algorithm sequence of applications reduction rules.
- Applying a α-lossy reduction rule ℓ times leads to a α^ℓ loss.

$$(I, k) \iff (I_1, k_1) \iff (I_2, k_2) \iff \ldots\ldots \iff (I_\ell, k_\ell)$$

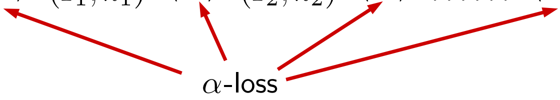
α-loss

- We modify the definition to allow for repeated application

Safe Reduction Rules

- A kernelization algorithm sequence of applications reduction rules.
- Applying a α-lossy reduction rule ℓ times leads to a α^ℓ loss.

$$(I, k) \iff (I_1, k_1) \iff (I_2, k_2) \iff \ldots\ldots \iff (I_\ell, k_\ell)$$



- We modify the definition to allow for repeated application

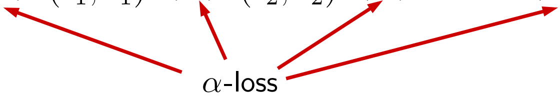
Each (Reduction rule, Solution lifting algorithm) pair satisfies

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

Safe Reduction Rules

- A kernelization algorithm sequence of applications reduction rules.
- Applying a α-lossy reduction rule ℓ times leads to a α^ℓ loss.

$$(I,k) \iff (I_1,k_1) \iff (I_2,k_2) \iff \ldots\ldots \iff (I_\ell,k_\ell)$$



- We modify the definition to allow for repeated application

Each (Reduction rule, Solution lifting algorithm) pair satisfies

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

This is called an α-safe reduction rule.

Safe Reduction Rules

- A kernelization algorithm sequence of applications reduction rules.
- Applying a α-lossy reduction rule ℓ times leads to a α^{ℓ} loss.

$$(I, k) \iff (I_1, k_1) \iff (I_2, k_2) \iff \ldots\ldots \iff (I_\ell, k_\ell)$$

α-loss

- We modify the definition to allow for repeated application

Each (Reduction rule, Solution lifting algorithm) pair satisfies

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

α-safe reduction rule.

We can chain α-safe reduction rules safely :)

Safe Reduction Rules

- A kernelization algorithm sequence of applications reduction rules.
- Applying a α-lossy reduction rule ℓ times leads to a α^ℓ loss.

$$(I,k) \iff (I_1,k_1) \iff (I_2,k_2) \iff \ldots\ldots \iff (I_\ell,k_\ell)$$

α-loss

- We modify the definition to allow for repeated application

Each (Reduction rule, Solution lifting algorithm) pair satisfies

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

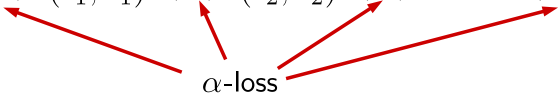
α-safe reduction rule.

Solution quality is always α or better !

Safe Reduction Rules

- A kernelization algorithm sequence of applications reduction rules.
- Applying a α-lossy reduction rule ℓ times leads to a α^ℓ loss.

$$(I,k) \iff (I_1,k_1) \iff (I_2,k_2) \iff \ldots\ldots \iff (I_\ell,k_\ell)$$



- We modify the definition to allow for repeated application

Each (Reduction rule, Solution lifting algorithm) pair satisfies

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

α-safe reduction rule.

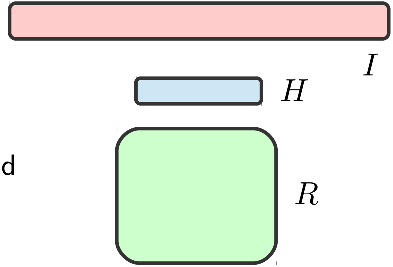
Solution quality is always α or better !

Assuming that the kernel-solution was quality α or better

Connected Vertex Cover

Input: A graph G *and a number* k

Question: Is there a vertex cover of value k that is also connected ?

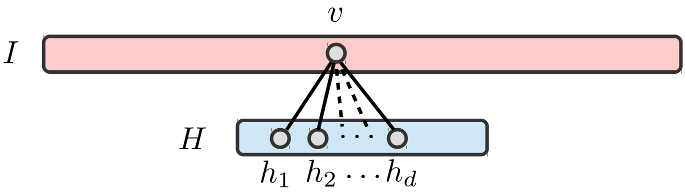


- H : vertices of degree $\geq k+1$
- I : vertices whose neighborhood is contained in H
- R : the remaining vertices

Connected Vertex Cover

Input: A graph G *and a number* k

Question: Is there a vertex cover of value k that is also connected ?



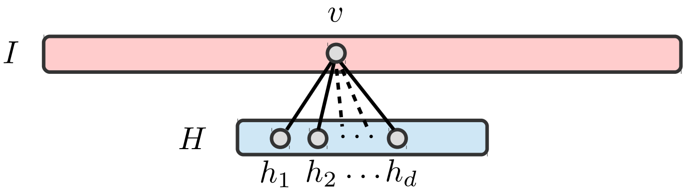
If there here is a vertex of degree $\geq d$ in I

Connected Vertex Cover

Input: A graph G *and a number* k

Question: Is there a vertex cover of value k that is also connected ?

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



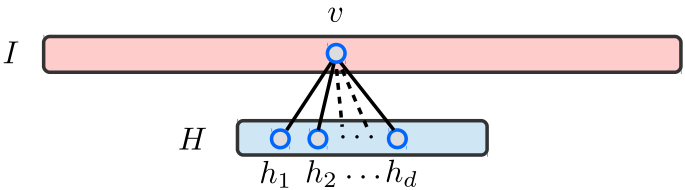
If there here is a vertex of degree $\geq d$ in I

Connected Vertex Cover

Input: A graph G *and a number* k

Question: Is there a vertex cover of value k that is also connected ?

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



Reduction Rule :

If there is a vertex $v \in I$ that has d neighbors (in H), then contract $\{v, h_1, h_2, \ldots, h_d\}$ into a single vertex $w \in H$ (by add $k+1$ new neighbors)

$$k' \leftarrow k - d + 1$$

Connected Vertex Cover

Input: A graph G *and a number* k

Question: Is there a vertex cover of value k that is also connected ?

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

Solution Lifting Algorithm :

Return $S = S' - w \cup \{v, h_1, \ldots, h_d\}$

Connected Vertex Cover

Input: A graph G *and a number* k

Question: Is there a vertex cover of value k that is also connected ?

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

Solution Lifting Algorithm :

Return $S = S' - w \cup \{v, h_1, \ldots, h_d\}$

No loss of connectivity

But may use 1 extra vertex

Connected Vertex Cover

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

Input: A graph G *and a number* k

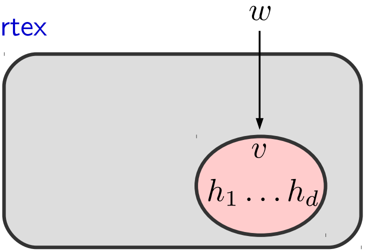
Question: Is there a vertex cover of value k that is also connected ?

Solution Lifting Algorithm :

Return $S = S' - w \cup \{v, h_1, \ldots, h_d\}$

No loss of connectivity

But may use 1 extra vertex



Connected Vertex Cover

Input: A graph G *and a number* k

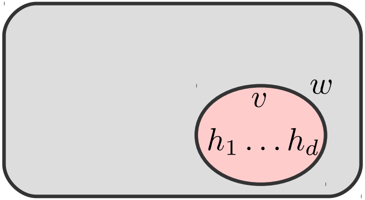
Question: Is there a vertex cover of value k that is also connected ?

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

Solution Lifting Algorithm :

Return $S = S' - w \cup \{v, h_1, \ldots, h_d\}$

This is α-safe



Connected Vertex Cover

Input: A graph G *and a number* k

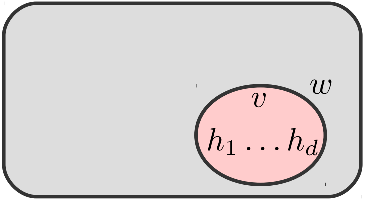
Question: Is there a vertex cover of value k that is also connected ?

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

Solution Lifting Algorithm :

Return $S = S' - w \cup \{v, h_1, \ldots, h_d\}$

Claim 1: $OPT(G', k') \leq OPT(G, k) - d + 1$



Connected Vertex Cover

Input: A graph G *and a number* k
Question: Is there a vertex cover of value k that is also connected ?

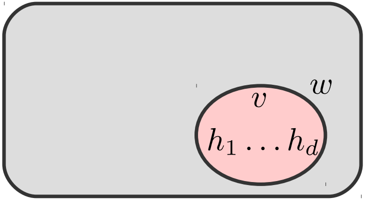
$$d = \lceil \frac{\alpha}{\alpha - 1} \rceil$$

Solution Lifting Algorithm :

Return $S = S' - w \cup \{v, h_1, \ldots, h_d\}$

Claim 1: $OPT(G', k') \leq OPT(G, k) - d + 1$

If $\tilde{S}$ is an optimum solution set for (G, k)
Then $S' = \tilde{S} / \{v, h_1, \ldots, h_d\} + w$
is a solution set for (G', k')



Connected Vertex Cover

Input: A graph G *and a number* k

Question: Is there a vertex cover of value k that is also connected ?

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

Solution Lifting Algorithm :

Return $S = S' - w \cup \{v, h_1, \ldots, h_d\}$

Claim 1: $OPT(G', k') \leq OPT(G, k) - d + 1$

Claim 2: $CVC(G, k, S) \leq CVC(G', k', S') + d$

Connected Vertex Cover

Input: A graph G *and a number* k

Question: Is there a vertex cover of value k that is also connected ?

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

Solution Lifting Algorithm :

Return $S = S' - w \cup \{v, h_1, \ldots, h_d\}$

Claim 1: $OPT(G', k') \leq OPT(G, k) - d + 1$

Claim 2: $CVC(G, k, S) \leq CVC(G', k', S') + d$

Remove 1 vertex and add d + 1

Connected Vertex Cover

Input: A graph G *and a number* k

Question: Is there a vertex cover of value k that is also connected ?

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

Solution Lifting Algorithm :

Return $S = S' - w \cup \{v, h_1, \ldots, h_d\}$

Claim 1: $OPT(G', k') \leq OPT(G, k) - d + 1$

Claim 2: $CVC(G, k, S) \leq CVC(G', k', S') + d$

$$\frac{CVC(G,k,S)}{OPT(G,k)} \leq \frac{CVC(G',k',S') + d}{OPT(G',k') + (d-1)} \leq max\Big\{\frac{CVC(G',k',S')}{OPT(G',k')}, \alpha\Big\}$$

Connected Vertex Cover

Input: A graph G *and a number* k

Question: Is there a vertex cover of value k that is also connected ?

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

Solution Lifting Algorithm :

Return $S = S' - w \cup \{v, h_1, \ldots, h_d\}$

Claim 1: $OPT(G', k') \leq OPT(G, k) - d + 1$

Claim 2: $CVC(G, k, S) \leq CVC(G', k', S') + d$

$$\frac{CVC(G,k,S)}{OPT(G,k)} \leq \frac{CVC(G',k',S') + d}{OPT(G',k') + (d-1)} \leq max\left\{\frac{CVC(G',k',S')}{OPT(G',k')}, \alpha\right\}$$

$$\frac{a+x}{b+y} \leq max\{\frac{a}{b}, \frac{x}{y}\}$$

Connected Vertex Cover

Input: A graph G *and a number* k

Question: Is there a vertex cover of value k that is also connected ?

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

Solution Lifting Algorithm :

Return $S = S' - w \cup \{v, h_1, \ldots, h_d\}$

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

Claim 1: $OPT(G', k') \leq OPT(G, k) - d + 1$

Claim 2: $CVC(G, k, S) \leq CVC(G', k', S') + d$

$$\frac{CVC(G,k,S)}{OPT(G,k)} \leq \frac{CVC(G',k',S') + d}{OPT(G',k') + (d-1)} \leq max\left\{\frac{CVC(G',k',S')}{OPT(G',k')}, \alpha\right\}$$

$$\frac{a+x}{b+y} \leq max\{\frac{a}{b}, \frac{x}{y}\}$$

Connected Vertex Cover

Input: A graph G *and a number* k

Question: Is there a vertex cover of value k that is also connected ?

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

Solution Lifting Algorithm :

Return $S = S' - w \cup \{v, h_1, \ldots, h_d\}$

Claim 1: $OPT(G', k') \leq OPT(G, k) - d + 1$

Claim 2: $CVC(G, k, S) \leq CVC(G', k', S') + d$

$$\frac{CVC(G,k,S)}{OPT(G,k)} \leq \frac{CVC(G',k',S') + d}{OPT(G',k') + (d-1)} \leq max\left\{\frac{CVC(G',k',S')}{OPT(G',k')}, \alpha\right\}$$

Hence Reduction Rule 2 is α-safe

Connected Vertex Cover

Input: A graph G *and a number* k

Question: Is there a vertex cover of value k that is also connected ?

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

Reduction Rule :

Remove any vertex in I with more than $k+1$ twins

Connected Vertex Cover

Input: A graph G *and a number* k

Question: Is there a vertex cover of value k that is also connected ?

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

Every vertex in I must have a neighbor in H

And any $h_1 \ h_2 \cdots h_d$ has no common neighbor in I

Connected Vertex Cover

Input: A graph G *and a number* k

Question: Is there a vertex cover of value k that is also connected ?

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

Every vertex in I must have a neighbor in H

And any $h_1 \ h_2 \cdots h_d$ has no common neighbor in I

So any vertex in I has degree $\leq d - 1$

Connected Vertex Cover

Input: A graph G *and a number* k
Question: Is there a vertex cover of value k that is also connected ?

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

Every vertex in I must have a neighbor in H
And any $h_1 \ h_2 \cdots h_d$ has no common neighbor in I

So any vertex in I has degree $\leq d - 1$

The size of G' is bounded by $\mathcal{O}(k^d + k^2)$

Connected Vertex Cover

Input: A graph G *and a number* k
Question: Is there a vertex cover of value k that is also connected ?

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

Every vertex in I must have a neighbor in H
And any $h_1 \ h_2 \cdots h_d$ has no common neighbor in I

So any vertex in I has degree $\leq d - 1$

The size of G' is bounded by $\mathcal{O}(k^d + k^2)$

Hence a Lossy Polynomial Kernel for CVC !

Thank you