Lossy Kernelization

Roohani Sharma

Slides Courtesy: Saket Saurabh

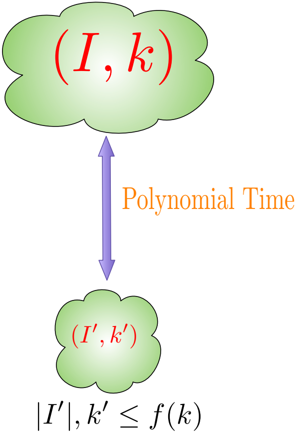
Lecture #14 Contd.
February 08, 2022

Mantra

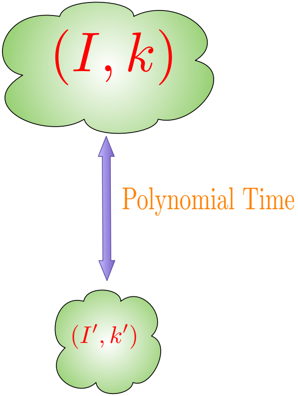
Ideally, one should be able to run a pre-processing algorithm before running any algorithm, parameterized/approximation/heuristics, on the input.

An important drawback!

But the notion of kernelization, as we have been seeing, does not combine well with approximation algorithms or with heuristics.



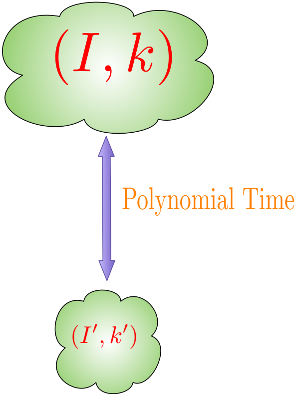
Why?



$|I'|, k' \leq f(k)$

Why?

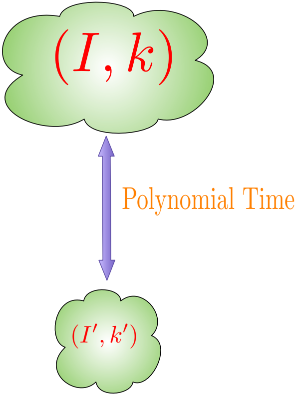
- Run 2-approximation on (I', k') and get solution S.



$|I'|, k' \leq f(k)$

Why?

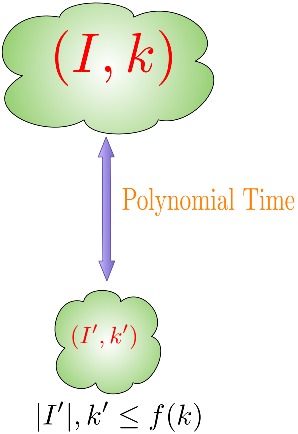
- Run 2-approximation on (I', k') and get solution S.
- Can we use S to get solution for I?



$|I'|, k' \leq f(k)$

Why?

- Can we use S to get solution for I?
- The current definition provides no insight whatsoever about the original instance.



Why?

- If we have an α-approximate solution to (I', k') there is no guarantee that we will be able to get an α-approximate solution to (I, k), or even able to get any feasible solution to (I, k).

Some Remarks

- In practice most kernels are okay.

Some Remarks

- In practice most kernels are okay.

It is primarily a limitation of the definition.

Definition is broken :(

Let us fix it.

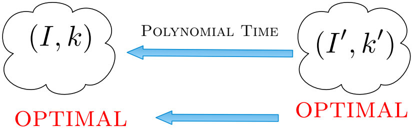
Definition is broken :(

$$(I, k) \xleftrightarrow{\text{Polynomial Time}} (I', k')$$

$$\text{Yes} \longleftrightarrow \text{Yes}$$

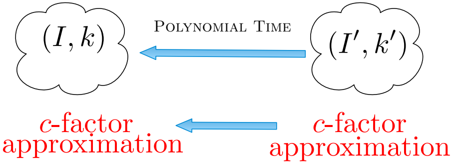
$$\text{optimal} \longleftrightarrow \text{optimal}$$

Definition is broken :(



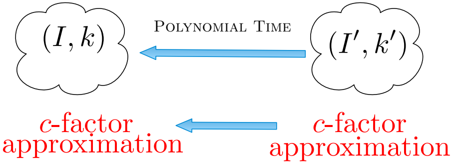
Useful information in reduced instance gives useful information in the original instance.

Kernel?



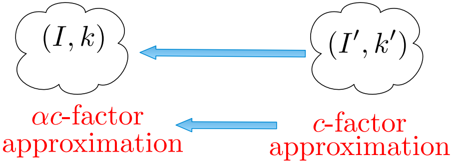
Observe that the inequality should hold for all values of c.

Kernel?



Observe that the inequality should hold for all values of c.
If allowing loss in reduced instance why not allow loss in reduction itself!

α-Approximate/Lossy Kernel (Rough Version)



Observe that the inequality should hold for all values of c.

Solution for reduced instance $\implies$ α bad solution for original

Technicalities

Approximation is about optimization problems.

Technicalities

Approximation is about optimization problems.

Most of the parameterized/kernelization algorithms are built around decision problems.

Technicalities

Approximation is about optimization problems.

Most of the parameterized/kernelization algorithms are built around decision problems.

Need a notion of parameterized optimization problems.

Parameterized Optimization Problems

Definition

A parameterized optimization (minimization or maximization) problem Π is a computable function

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

Parameterized Optimization Problems

Definition

A parameterized optimization (minimization or maximization) problem Π is a computable function

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

- Instances of a parameterized optimization problem Π are pairs $(I, k) \in \Sigma^* \times \mathbb{N}$.

Parameterized Optimization Problems

Definition

A parameterized optimization (minimization or maximization) problem Π is a computable function

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

- Instances of a parameterized optimization problem Π are pairs $(I, k) \in \Sigma^* \times \mathbb{N}$.
- A *solution* to (I, k) is simply a string $s \in \Sigma^*$, such that $|s| \leqslant |I| + k$.

Parameterized Optimization Problems

Definition

A parameterized optimization (minimization or maximization) problem Π is a computable function

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

- Instances of a parameterized optimization problem Π are pairs $(I, k) \in \Sigma^* \times \mathbb{N}$.
- A *solution* to (I, k) is simply a string $s \in \Sigma^*$, such that $|s| \leqslant |I| + k$.
- The *value* of the solution s is $\Pi(I, k, s)$.

- Just as for “classical” optimization problems the instances of Π are given as input, and the algorithmic task is to find a solution with the best possible value.
- Best means minimization and maximization.

- Just as for “classical” optimization problems the instances of Π are given as input, and the algorithmic task is to find a solution with the best possible value.
- Best means minimization and maximization.
- *So we need a notion of optimum for parameterized optimization problems.*

Optimum Value

Definition

For a parameterized minimization problem Π, the *optimum value* of an instance $(I, k) \in \Sigma^* \times \mathbb{N}$ is

$$OPT_\Pi(I, k) = \min_{\substack{s \in \Sigma^* \\ |s| \leqslant |I| + k}} \Pi(I, k, s).$$

Optimum Value

Definition

For a parameterized minimization problem Π, the *optimum value* of an instance $(I, k) \in \Sigma^* \times \mathbb{N}$ is

$$OPT_\Pi(I, k) = \min_{\substack{s \in \Sigma^* \\ |s| \leqslant |I| + k}} \Pi(I, k, s).$$

For an instance (I, k) of a parameterized optimization problem Π, an *optimal solution* is a solution s such that $\Pi(I, k, s) = OPT_\Pi(I, k)$.

Optimum Value

Definition
For a parameterized minimization problem Π, the *optimum value* of an instance $(I, k) \in \Sigma^* \times \mathbb{N}$ is

$$OPT_\Pi(I, k) = \min_{\substack{s \in \Sigma^* \\ |s| \leqslant |I| + k}} \Pi(I, k, s).$$

For an instance (I, k) of a parameterized optimization problem Π, an *optimal solution* is a solution s such that $\Pi(I, k, s) = OPT_\Pi(I, k)$.
For parameterized optimization problems the algorithm has to produce an optimal solution.

Example: Connected Vertex Cover

Connected Vertex Cover (CVC) **Parameter:** k
Input: A graph $G = (V, E)$ and a positive integer k.
Question: Does there exist a subset $V' \subseteq V$ of size at most k such that V' is a vertex cover and $G[V']$ is connected?

Example with Connected Vertex Cover

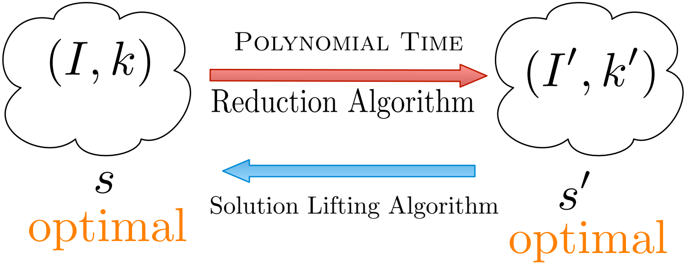
$$CVC(G,k,S) = \begin{cases} \infty & \text{if } S \text{ is not a cvc of the graph } G \\ |S| & \text{otherwise} \end{cases}$$

Example with Connected Vertex Cover

$$CVC(G,k,S) = \begin{cases} \infty & \text{if } S \text{ is not a cvc of the graph } G \\ |S| & \text{otherwise} \end{cases}$$

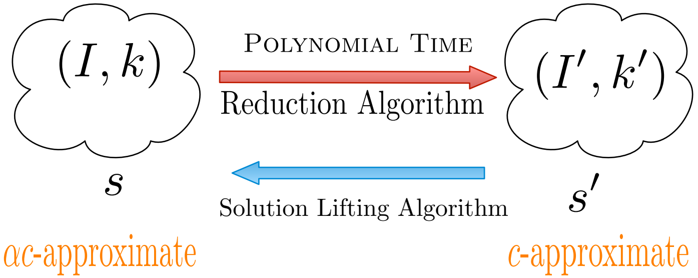
$$CVC(G,k,S) = \begin{cases} \infty & \text{if } S \text{ is not a cvc of the graph } G \\ \min\{|S|, k+1\} & \text{otherwise} \end{cases}$$

Kernels for Parameterized Optimization Problems



Kernelization comprises of a pair of algorithms: Reduction Algorithm and Solution Lifting Algorithm

α-Lossy Kernels



Which of the problems that do not admit polynomial kernel, admit α-lossy polynomial kernel?

Connected Vertex Cover

Connected Vertex Cover

What are the right question for this problem in this framework?

- It has a factor 2-approximation algorithm (that runs in polynomial time) $\implies$ $\mathcal{O}(1)$-sized 2-lossy kernel.

Connected Vertex Cover

What are the right question for this problem in this framework?

- It has a factor 2-approximation algorithm (that runs in polynomial time) $\implies$ $\mathcal{O}(1)$-sized 2-lossy kernel.
- It has no polynomial kernel $\implies$ no $k^{\mathcal{O}(1)}$-sized 1-lossy kernel.

Connected Vertex Cover

What are the right question for this problem in this framework?

- It has a factor 2-approximation algorithm (that runs in polynomial time) $\implies$ $\mathcal{O}(1)$-sized 2-lossy kernel.
- It has no polynomial kernel $\implies$ no $k^{\mathcal{O}(1)}$-sized 1-lossy kernel.
- So the right question is: does this has α-lossy kernel of $k^{\mathcal{O}(1)}$-size where $1 < \alpha < 2$.

What is the best answer one can hope for?

Connected Vertex Cover

What is the best answer one can hope for?

- For every $\alpha > 1$, it has α-approximate kernel of $k^{f(\alpha)}$ -size.

How to design α-lossy kernels

How to design α-lossy kernels

- We have the notion of safe Reduction Rules for kernelization.
- (I, k) if and only if (I', k').

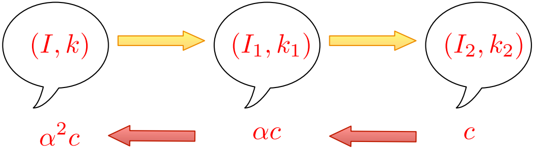
$$(I, k) \iff (I_1, k_1) \iff (I_2, k_2) \cdots \iff (I_\ell, k_\ell)$$

Designing α-lossy kernel

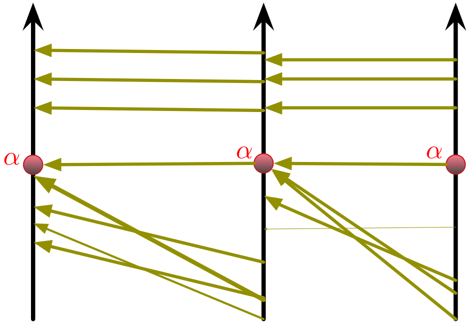
- Let us define an analogous notion of α**-Safe Reduction Rules** for α-lossy kernelization.
- $(I, k) \implies (I', k')$. Given a c-approximate solution to (I', k'), one should get an αc approximate solution for (I, k).

Designing α-lossy kernel

- Let us define an analogous notion of **α-Safe Reduction Rules** for α-lossy kernelization.
- $(I, k) \implies (I', k')$. Given a c-approximate solution to (I', k'), one should get an αc approximate solution for (I, k).

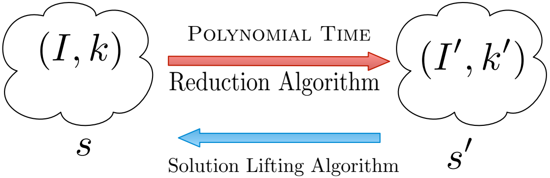


α-Safe Reduction Rule



- s' be a c-approximate solution to (I', k')
- If $c \leqslant \alpha$ then s must be at most α approximate solution to (I, k).
- If $c > \alpha$ then s must be at most c approximate solution to (I, k).

α-Safe Reduction Rule



If Π is a minimization problem

then

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

Towards α-lossy kernel for CONNECTED VERTEX COVER (CVC)

Recall the reduction rules for kernelization of VERTEX COVER.

Reduction Rule

Delete isolated vertices.

Reduction Rule (High degree rule)

Delete a vertex of degree at least $k + 1$ *and pick it in the solution.*

We cannot apply Reduction Rule 2.2 for CONNECTED VERTEX COVER.

Exponential Kernel for CVC

Let H be the set of vertices of degree at least $k+1$ in the graph. We know H is present in every at most k-sized solution. Thus, if $|H| \geqslant k+1$, report No-instance. Otherwise, $|H| \leqslant k$.

Exponential Kernel for CVC

Let H be the set of vertices of degree at least $k+1$ in the graph. We know H is present in every at most k-sized solution. Thus, if $|H| \geqslant k+1$, report No-instance. Otherwise, $|H| \leqslant k$.

If there are at least k^2+1 edges in $G-H$, then report No-instance.
Let R be the set of vertices in $G-H$ that are incident on some edge of $G-H$. Then $|R| \leqslant 2k^2$.

Exponential Kernel for CVC

Let H be the set of vertices of degree at least $k+1$ in the graph. We know H is present in every at most k-sized solution. Thus, if $|H| \geqslant k+1$, report No-instance. Otherwise, $|H| \leqslant k$.

If there are at least k^2+1 edges in $G-H$, then report No-instance.
Let R be the set of vertices in $G-H$ that are incident on some edge of $G-H$. Then $|R| \leqslant 2k^2$.

Let I be the remaining vertices of G, that is $I = V(G)\backslash(H \cup R)$.
Then $N(I) \subseteq H$.
The role of I is only to provide connectivity amongst the vertices of H.
We want to bound the size of I.

Exponential kernel for CVC

Reduction Rule

Reduction algorithm: *If two vertices of I have the same neighbourhood, then delete one of them.*

Solution lifting algorithm: *Every solution of the reduced instance is a solution of the original instance. In particular, an optimum solution of the reduced instance is also an optimum solution of the original instance.*

Reduction rule 2.3 is 1-safe. Reduction rule 2.3 implies $|I| \leqslant 2^{|H|} \leqslant 2^k$. This would immediately give an exponential kernel.

α-lossy kernel for CVC

Let d be the least integer such that $\frac{d}{d-1} \leqslant \alpha$. In particular, choose $d = \lceil \frac{\alpha}{\alpha-1} \rceil$.

Reduction Rule

Reduction algorithm: *Let* $v \in I$ *be a vertex of degree* $D \geqslant d$. *Delete* $N_G[v]$ *from* G *and add a vertex* w *such that the neighborhood of* w *is* $N_G(N_G(v)) \backslash \{v\}$. *Then add* k *degree* 1 *vertices* $v_1, \ldots, v_k$ *whose neighbor is* w. *Output this graph* G', *together with the new parameter* $k' = k - (D-1)$.

Solution lifting algorithm: *Return* $S = (S' \backslash \{w\}) \cup N_G[v]$, *where* S' *is a solution for* (G', k').

α-lossy kernel for CVC

Let d be the least integer such that $\frac{d}{d-1} \leqslant \alpha$. In particular, choose $d = \lceil \frac{\alpha}{\alpha-1} \rceil$.

Reduction Rule

Reduction algorithm: *Let $v \in I$ be a vertex of degree $D \geqslant d$. Delete $N_G[v]$ from G and add a vertex w such that the neighborhood of w is $N_G(N_G(v)) \backslash \{v\}$. Then add k degree 1 vertices $v_1, \ldots, v_k$ whose neighbor is w. Output this graph G', together with the new parameter $k' = k - (D-1)$.*
Solution lifting algorithm: *Return $S = (S' \backslash \{w\}) \cup N_G[v]$, where S' is a solution for (G', k').*

To prove: Reduction rule 2.4 is α-safe, that is, $\frac{CVC(I,k,S)}{OPT(I,k)} \leqslant \max\{\frac{CVC(I',k',S')}{OPT(I',k')}, \alpha\}$.

α-lossy kernel for CVC: Proof of Reduction rule 2.4

To prove: $\frac{CVC(I,k,S)}{OPT(I,k)} \leqslant \max\{\frac{CVC(I',k',S')}{OPT(I',k')}, \alpha\}$.

$$\boxed{OPT(G',k') \leqslant OPT(G,k) - (D-1)}$$

If $\tilde{S}$ is an optimum solution for (G,k), then
$S^* = \tilde{S} \backslash N_G(v) \cup \{w\}$ is a solution for (G',k').

α-lossy kernel for CVC: Proof of Reduction rule 2.4

To prove: $\frac{CVC(I,k,S)}{OPT(I,k)} \leqslant \max\{\frac{CVC(I',k',S')}{OPT(I',k')}, \alpha\}$.

$$\boxed{OPT(G',k') \leqslant OPT(G,k) - (D-1)}$$

If $\tilde{S}$ is an optimum solution for (G,k), then
$S^* = \tilde{S} \backslash N_G(v) \cup \{w\}$ is a solution for (G',k').

$$\boxed{CVC(G,k,S) \leqslant CVC(G',k',S') + D}$$

This follows from the construction of S ($S = (S' \backslash \{w\}) \cup N_G[v]$).

α-lossy kernel for CVC: Proof of Reduction rule 2.4

To prove: $\frac{CVC(I,k,S)}{OPT(I,k)} \leqslant \max\{\frac{CVC(I',k',S')}{OPT(I',k')}, \alpha\}$.

$$\boxed{OPT(G',k') \leqslant OPT(G,k) - (D-1)}$$

If $\tilde{S}$ is an optimum solution for (G,k), then
$S^* = \tilde{S} \backslash N_G(v) \cup \{w\}$ is a solution for (G',k').

$$\boxed{CVC(G,k,S) \leqslant CVC(G',k',S') + D}$$

This follows from the construction of S ($S = (S' \backslash \{w\}) \cup N_G[v]$).

$$\frac{CVC(I,k,S)}{OPT(I,k)} \leqslant \frac{CVC(I',k',S') + D}{OPT(I',k') - (D-1)} \leqslant \max\{\frac{CVC(I',k',S')}{OPT(I',k')}, \frac{D}{D-1}\}$$
$$\leqslant \max\{\frac{CVC(I',k',S')}{OPT(I',k')}, \alpha\}$$

Second inequality follows because because $\max\{\frac{a+x}{b+y} \leqslant \max\{\frac{a}{b}, \frac{x}{y}\}\}$.

α-lossy kernel for CVC: Size bound

When Reduction rule 2.3 and 2.4 are not applicable, the degree of each vertex of I is at most d and thus,
$|I| \leqslant \binom{|H|}{\leqslant d} = O(k^d) = O(k^{\lceil \frac{\alpha}{\alpha-1} \rceil})$.

α-lossy kernel for CVC: Size bound

When Reduction rule 2.3 and 2.4 are not applicable, the degree of each vertex of I is at most d and thus,
$|I| \leqslant \binom{|H|}{\leqslant d} = O(k^d) = O(k^{\lceil \frac{\alpha}{\alpha-1} \rceil})$.
Since $V(G) = H \uplus R \uplus I$, $|H| \leqslant k$, $|R| \leqslant 2k^2$, and $|I| = O(k^d)$, we get an α-lossy kernel for Connected Vertex Cover of size $O(k^{\lceil \frac{\alpha}{\alpha-1} \rceil} + k^2)$, for every $\alpha > 1$.

Further remarks on lossy kernels

1. Unlike Turing kernels, lossy kernels can also be designed for W-hard problems.
2. Unlike Turing kernels, there is a lower bound machinery that shows that no lossy kernels exist for certain problems under reasonable complexity theoretic assumptions. This theory is developed over the ideas from OR-compositions (a tool from kernelization lower bound) and gap creating reductions (a tool from approximation lower bound).