Lossy Kernelization

Daniel Lokshtanov[1] Fahad Panolan[2] M. S. Ramanujan[3]
Saket Saurabh[1,2]

[1]University of Bergen, Norway.
[2]The Institute of Mathematical Sciences, India
[3]Technische Universität Wien

Simon Institute, November 5th, 2015.

Introduction and Kernelization

Fixed Parameter Tractable (FPT) Algorithms

For decision problems with input size n, and a parameter k, (which typically is the solution size), the goal here is to design an algorithm with running time $f(k) \cdot n^{\mathcal{O}(1)}$, where f is a computable function of k alone.

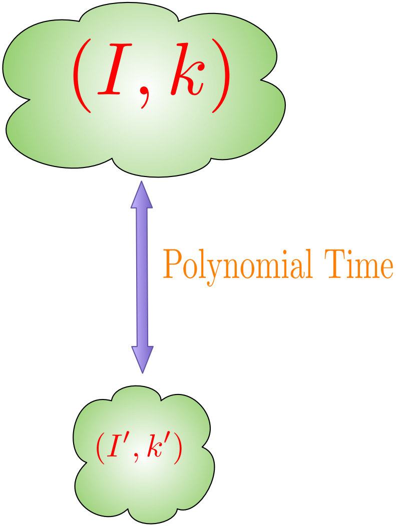
Problems that have such an algorithm are said to be fixed parameter tractable (FPT).

A Few Examples

VERTEX COVER **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 for every edge $(u, v) \in E$ either $u \in V'$ or $v \in V'$?

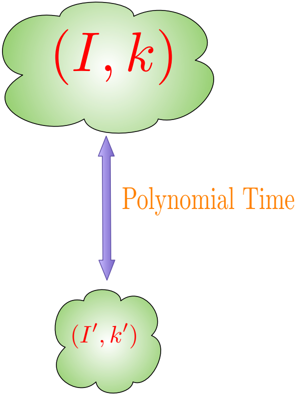
PATH **Parameter:** k
Input: A graph $G = (V, E)$ and a positive integer k.
Question: Does there exist a path P in G of length at least k?

Kernelization: A Method for Everyone



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

Kernelization: A Method for Everyone



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

INFORMALLY: A kernelization algorithm is a polynomial-time transformation that transforms any given parameterized instance to an equivalent instance of the same problem, with size and parameter bounded by a function of the parameter.

Kernel: Formally

FORMALLY: A kernelization algorithm, or in short, a kernel for a parameterized problem $L \subseteq \Sigma^* \times \mathbb{N}$ is an algorithm that given $(x, k) \in \Sigma^* \times \mathbb{N}$, outputs in $p(|x| + k)$ time a pair $(x', k') \in \Sigma^* \times \mathbb{N}$ such that

Kernel: Formally

FORMALLY: A kernelization algorithm, or in short, a kernel for a parameterized problem $L \subseteq \Sigma^* \times \mathbb{N}$ is an algorithm that given $(x, k) \in \Sigma^* \times \mathbb{N}$, outputs in $p(|x| + k)$ time a pair $(x', k') \in \Sigma^* \times \mathbb{N}$ such that

- $(x, k) \in L \iff (x', k') \in L$,
- $|x'|, k' \leqslant f(k)$,

where f is an arbitrary computable function, and p a polynomial. Any function f as above is referred to as the size of the kernel.

Kernel: Formally

FORMALLY: A kernelization algorithm, or in short, a kernel for a parameterized problem $L \subseteq \Sigma^* \times \mathbb{N}$ is an algorithm that given $(x, k) \in \Sigma^* \times \mathbb{N}$, outputs in $p(|x| + k)$ time a pair $(x', k') \in \Sigma^* \times \mathbb{N}$ such that

- $(x, k) \in L \iff (x', k') \in L$,
- $|x'|, k' \leqslant f(k)$,

where f is an arbitrary computable function, and p a polynomial. Any function f as above is referred to as the size of the kernel.

Polynomial kernel $\implies$ f is polynomial.

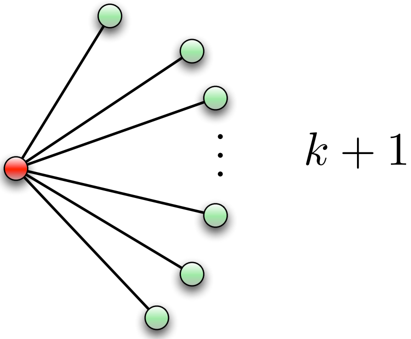
Example 1: Vertex Cover

Rule 1: Remove any isolated vertices.

Example 1: Vertex Cover

Rule 1: Remove any isolated vertices.

Rule 2: If there is a vertex v of degree at least $k+1$ then include v in solution and $(G - \{v\}, k-1)$



Example 1: Vertex Cover

Rule 1: Remove any isolated vertices.

Rule 2: If there is a vertex v of degree at least $k+1$ then include v in solution and $(G - \{v\}, k-1)$

Apply these rules until no longer possible.

Example 1: Vertex Cover

Rule 1: Remove any isolated vertices.

Rule 2: If there is a vertex v of degree at least $k+1$ then include v in solution and $(G - \{v\}, k-1)$

Apply these rules until no longer possible.
What conclusions can we draw ?

Example 1: Vertex Cover

Rule 1: Remove any isolated vertices.

Rule 2: If there is a vertex v of degree at least $k+1$ then include v in solution and $(G - \{v\}, k-1)$

Apply these rules until no longer possible.

What conclusions can we draw ?

Outcome 1: If G is not empty and k drops to 0 — the answer is No.

Example 1: VERTEX COVER

Rule 1: Remove any isolated vertices.

Rule 2: If there is a vertex v of degree at least $k+1$ then include v in solution and $(G - \{v\}, k-1)$

Apply these rules until no longer possible.
What conclusions can we draw ?

Outcome 1: If G is not empty and k drops to 0 — the answer is NO.

Observation: Every vertex has degree at most k — number of edges they can cover is at most k^2.

Example 1: VERTEX COVER

Rule 1: Remove any isolated vertices.

Rule 2: If there is a vertex v of degree at least $k+1$ then include v in solution and $(G - \{v\}, k-1)$

Apply these rules until no longer possible.
What conclusions can we draw ?

Outcome 1: If G is not empty and k drops to 0 — the answer is NO.

Observation: Every vertex has degree at most k — number of edges they can cover is at most k^2.

Outcome 2: If $|E| > k^2$ — the answer is NO. Else $|E| \leqslant k^2$, $|V| \leqslant 2k^2$ and we have polynomial sized kernel of $\mathcal{O}(k^2)$.

A decidable problem admits a kernel if and only if it is fixed-parameter tractable.


2008.

Connected Vertex Cover

Odd Cycle Transversal

Vertex Cover

Disjoint Factor

Cycle Packing

Directed Feedback Vertex Set

Feedback Vertex Set

Multiway Cut

Steiner Tree

Path

Almost-2-SAT

Min-Ones-d-SAT

Chordal Vertex Deletion

Tree Isomorphism

A decidable problem admits a kernel if and only if it is fixed-parameter tractable.

A decidable problem admits a kernel if and only if it is fixed-parameter tractable.

A fine grained question: Which of these problems admit polynomial kernel?.

Theory of Lower Bound

- Hans L. Bodlaender, Rodney G. Downey, Michael R. Fellows, Danny Hermelin: On Problems without Polynomial Kernels (Extended Abstract). ICALP (1) 2008: 563-574
- Lance Fortnow, Rahul Santhanam: Infeasibility of instance compression and succinct PCPs for NP. STOC 2008: 133-142
- Holger Dell, Dieter van Melkebeek: Satisfiability allows no nontrivial sparsification unless the polynomial-time hierarchy collapses. STOC 2010: 251-260

Theory of Lower Bound

- Hans L. Bodlaender, Rodney G. Downey, Michael R. Fellows, Danny Hermelin: On Problems without Polynomial Kernels (Extended Abstract). ICALP (1) 2008: 563-574
- Lance Fortnow, Rahul Santhanam: Infeasibility of instance compression and succinct PCPs for NP. STOC 2008: 133-142
- Holger Dell, Dieter van Melkebeek: Satisfiability allows no nontrivial sparsification unless the polynomial-time hierarchy collapses. STOC 2010: 251-260

Some problems do not admit polynomial kernel unless $\mathrm{NP} \subseteq \frac{\text{co-NP}}{\text{Poly}}$

No Poly Kernel

CONNECTED VERTEX COVER

DISJOINT FACTOR

CYCLE PACKING

STEINER TREE

PATH

MIN-ONES-d-SAT

TREE ISOMORPHISM

Poly Kernel

ODD CYCLE TRANSVERSAL

VERTEX COVER

FEEDBACK VERTEX SET

ALMOST-2-SAT

OPEN

MULTIWAY CUT

CHORDAL VERTEX DELETION

DIRECTED FEEDBACK VERTEX SET

Goal to clean the picture even more.

Goal to clean the picture even more.

That will have to wait a bit. First we take a detour.

It never hurts to run kernelization algorithm before running any algorithm such as approximation or heuristics!

Mantra

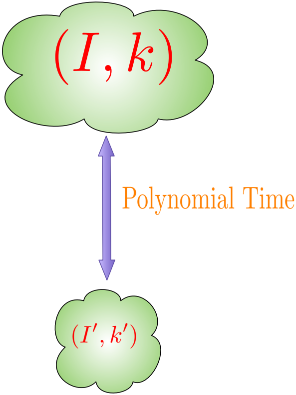
It never hurts to run kernelization algorithm before running any algorithm such as approximation or heuristics!

Really :).

An important drawback!

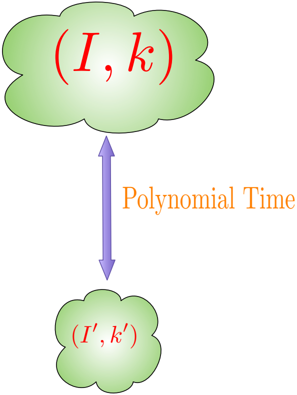
It does not combine well with approximation algorithms or with heuristics.

Why?



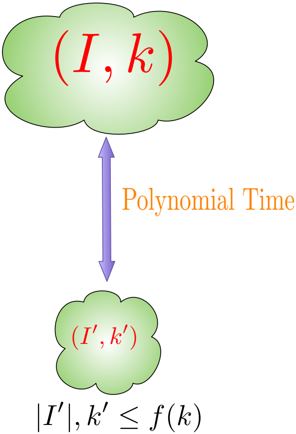
$|I'|, k' \leq f(k)$

Why?



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

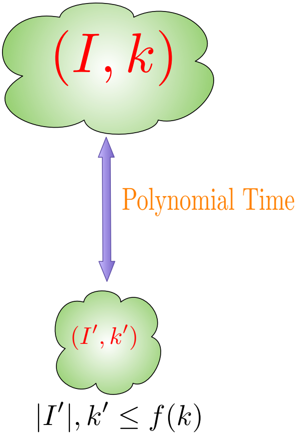
- Run 2-approximation on (I', k') and get solution S.



Why?

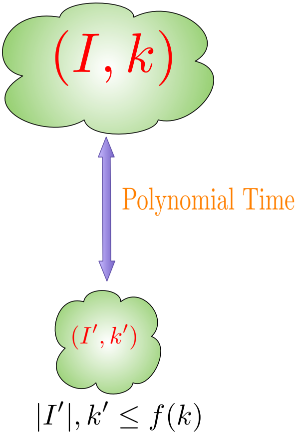
- Run 2-approximation on (I', k') and get solution S.
- Can we use S to get solution for I?

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.
- For example we could just run kernel with larger values of k.

Some Remarks

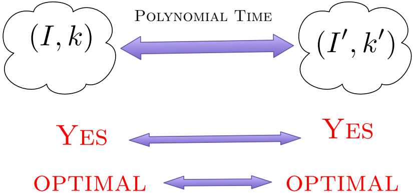
- In practice most kernels are okay.
- For example we could just run kernel with larger values of k.

It is primarily a limitation of the definition.

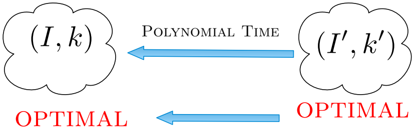
Definition is broken :(

Let us fix it.

Definition is broken :(

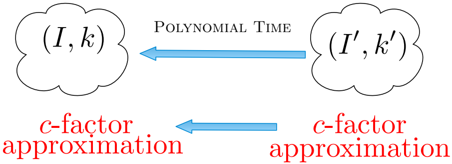


Definition is broken :(



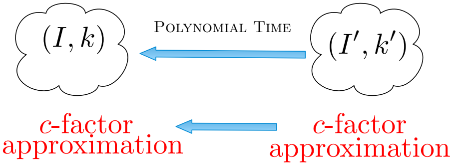
Useful information in reduced instance gives useful information in the original instance.

Kernel?



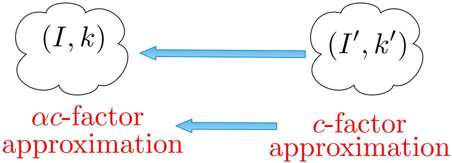
Observe that the inequality should hold for all values of c.

Kernel?



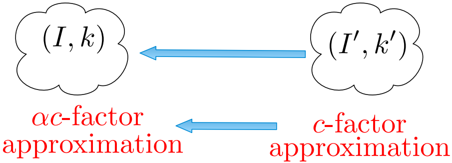
Observe that the inequality should hold for all values of c.
If allowing loss in reduced instance why not allow loss in reduction itself!

α-Approximate Kernel (Rough Version)



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

α-Approximate Kernel (Rough Version)



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

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

α-Approximate Kernel: Expectations

- Should fit with approximation world.
- Should fit with the fpt - approximations.

Why it seems difficult?

Approximation is about optimization problems.

Why it seems difficult?

Approximation is about optimization problems.

Most of Parameterized algorithms is built around decision problems.

Why it seems difficult?

Approximation is about optimization problems.

Most of Parameterized algorithms is built around decision problems.

Of course except doing Parameterized Approximation.

Need a notion of parameterized optimization problems.

Need a notion of parameterized optimization problems.

Build parameterized complexity with this notion.

Need a notion of parameterized optimization problems.

Build parameterized complexity with this notion.

Robust — Versatile – Natural

Need a notion of parameterized optimization problems.

Build parameterized complexity with this notion.

Robust — Versatile – Natural

Encompass both Parameterized Algorithms, Approximation Algorithms and Kernelization.

Parameterized optimization problems.

1. Yijia Chen, Martin Grohe, Magdalena Grüber: On Parameterized Approximability. IWPEC 2006: 109-120
2. Dániel Marx. Parameterized complexity and approximation algorithms. The Computer Journal, 51(1):60-78, 2008.

Parameterized optimization problems.

1. Yijia Chen, Martin Grohe, Magdalena Grüber: On Parameterized Approximability. IWPEC 2006: 109-120
2. Dániel Marx. Parameterized complexity and approximation algorithms. The Computer Journal, 51(1):60-78, 2008.

We could build the approximate-kernelization framework starting from these but we give a different definition and use that for our framework.

An attempt to build such a framework for approximate kernelization.

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

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

- 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

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

- 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

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

- 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 $\Pi(I, k, s) = OPT_\Pi(I, k)$.

Example with 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

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

Example with Connected Vertex Cover

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

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

Solving Parameterized Optimization Problems

Let Π be a parameterized optimization problem.

- Algorithm *solves* Π if, for every instance (I, k) the solution s output by the algorithm is optimal for (I, k).

Solving Parameterized Optimization Problems

Let Π be a parameterized optimization problem.

- Algorithm *solves* Π if, for every instance (I, k) the solution s output by the algorithm is optimal for (I, k).

Definition

A parameterized optimization problem Π is *fixed parameter tractable* (FPT) if there is an algorithm that solves Π, such that the running time of the algorithm on instances of size n with parameter k is upper bounded by $f(k)n^{\mathcal{O}(1)}$ for a computable function f.

Fixed Parameter Tractable (FPT) Algorithms

A parameterized decision problem Π is *fixed parameter tractable* (FPT) if there is an algorithm that decides Π, such that the running time of the algorithm on instances of size n with parameter k is upper bounded by $f(k)n^{\mathcal{O}(1)}$ for a computable function f.

A parameterized optimization problem Π is *fixed parameter tractable* (FPT) if there is an algorithm that solves Π, such that the running time of the algorithm on instances of size n with parameter k is upper bounded by $f(k)n^{\mathcal{O}(1)}$ for a computable function f.

- Solving a desicion problem amounts to always output “yes” on “yes”-instances and “no” on “no”-instances.

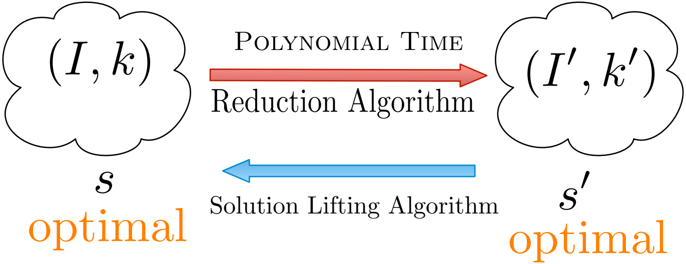
- Solving a desicion problem amounts to always output “yes” on “yes”-instances and “no” on “no”-instances.
- For parameterized optimization problems the algorithm has to produce an optimal solution.

- Solving a desicion problem amounts to always output "yes" on "yes"-instances and "no" on "no"-instances.
- For parameterized optimization problems the algorithm has to produce an optimal solution.
- This is analogous to the definition of optimization problems most commonly used in approximation algorithms.

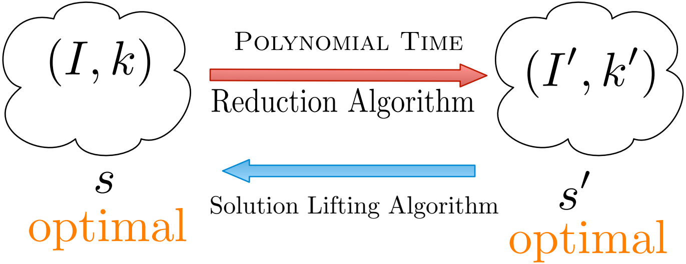
- Solving a desicion problem amounts to always output “yes” on “yes”-instances and “no” on “no”-instances.
- For parameterized optimization problems the algorithm has to produce an optimal solution.
- This is analogous to the definition of optimization problems most commonly used in approximation algorithms.

Parameterized decision problems $\iff$ Parameterized Optimization Problems

Kernels for Parameterized Optimization Problems



Kernels for Parameterized Optimization Problems



A decidable parameterized optimization problem Π is FPT if and only if it admits a kernel.

Definition

Polynomial time α-approximation algorithm for a parameterized optimization problem Π is an algorithm that takes as input an instance (I, k), runs in time $|I|^{\mathcal{O}(1)}$, and outputs a solution s such that $\Pi(I, k, s) \leqslant \alpha \cdot OPT(I, k)$ if Π is a minimization problem.

FPT-Approximation

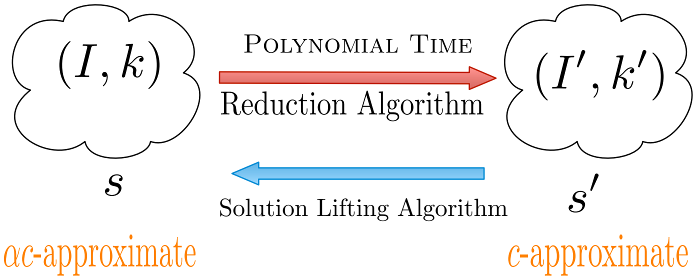
Definition

Polynomial time α-approximation algorithm for a parameterized optimization problem Π is an algorithm that takes as input an instance (I, k), runs in time $|I|^{\mathcal{O}(1)}$, and outputs a solution s such that $\Pi(I, k, s) \leqslant \alpha \cdot OPT(I, k)$ if Π is a minimization problem.

Definition

Let $\alpha \geqslant 1$ be constant. A fixed parameter tractable α-approximation algorithm for a parameterized optimization problem Π is an algorithm that takes as input an instance (I, k), runs in time $f(k)|I|^{\mathcal{O}(1)}$, and outputs a solution s such that $\Pi(I, k, s) \leqslant \alpha \cdot OPT(I, k)$ if Π is a minimization problem.

α-Approximate Kernel



α-Approximate Kernel and α-Approximation FPT Algorithm

For every $\alpha \geqslant 1$ and decidable parameterized optimization problem Π, Π admits a fixed parameter tractable α-approximation algorithm if and only if Π has an α-approximate kernel.

α-Approximate Kernel and α-Approximation FPT Algorithm

For every $\alpha \geqslant 1$ and decidable parameterized optimization problem Π, Π admits a fixed parameter tractable α-approximation algorithm if and only if Π has an α-approximate kernel.

For every $\alpha \geqslant 1$ and decidable parameterized optimization problem Π, Π admits a polynomial time α-approximation algorithm if and only if Π has an α-approximate kernel of constant size.

For every $\alpha \geqslant 1$ and decidable parameterized optimization problem Π, Π admits a fixed parameter tractable α-approximation algorithm if and only if Π has an α-approximate kernel.

A fine grained question: Which of these problems admit α-approximate polynomial kernel?

A fine grained question: Which of these problems admit α-approximate polynomial kernel?

In particular, which of the problems that do not admit polynomial kernel, admit α-approximate polynomial kernel.

Connected Vertex Cover

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

- It has a factor 2-approximation $\implies$ $\mathcal{O}(1)$-sized 2-approximate kernel.

Connected Vertex Cover

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

- It has a factor 2-approximation $\implies$ $\mathcal{O}(1)$-sized 2-approximate kernel.
- It has no polynomial kernel $\implies$ no $k^{\mathcal{O}(1)}$-sized 1-approximate kernel.

Connected Vertex Cover

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

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

Connected Vertex Cover

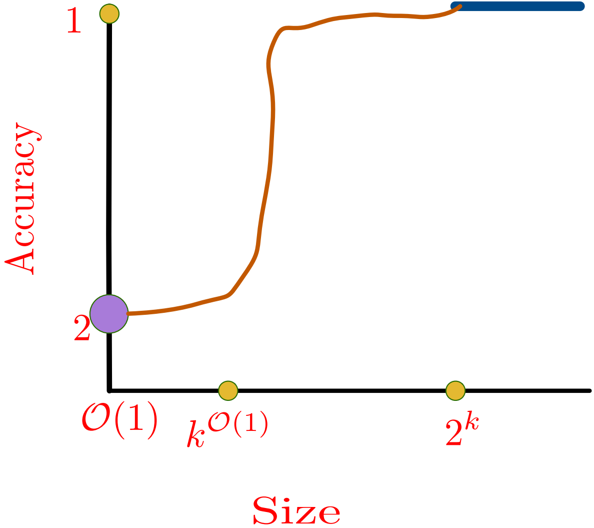
What is the best answer one can hope for?

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.

Connected Vertex Cover



Our Results

Problem Name	Apx.	Apx. Hardness	Kernel	Apx. Ker. Fact.	Appx. Ker. Size
VERTEX COVER	2 [44]	$(2-\epsilon)$ [17,33]	$2k$ [12]	$1<\alpha<2$	$2(2-\alpha)k$ [26]
d-HITTING SET	d [44]	$d-\epsilon$ [16,33]	$O(k^{d-1})$ [1]	$1<\alpha<d$	$O((k\cdot\frac{d-\alpha}{\alpha-1})^{d-1})$ [26]
STEINER TREE	1.39 [9]	no PTAS [10]	no $k^{O(1)}$ [18]	$1<\alpha$	$k^{f(\alpha)}$
OLA/V.C.	$O(\sqrt{\log n}\log\log n)$ [24]	no PTAS [3]	$f(k)$ [35]	$1<\alpha<2$	$f(\alpha)2^k k^4$
PARTIAL V.C.	$(\frac{4}{3}-\epsilon)$ [23]	no PTAS [38]	no $f(k)$ [29]	$1<\alpha$	$f(\alpha)k^5$
CONNECTED V.C.	2 [4,41]	$(2-\epsilon)$ [33]	no $k^{O(1)}$ [18]	$1<\alpha$	$k^{f(\alpha)}$
CYCLE PACKING	$O(\log n)$ [40]	$(\log n)^{\frac{1}{2}-\epsilon}$ [28]	no $k^{O(1)}$ [7]	6	$O((k\log k)^2)$
CYCLE PACKING				$1<\alpha$	$k^{f(\alpha)\log k}$
DISJOINT FACTORS	2	no PTAS	no $k^{O(1)}$ [7]	$1<\alpha$	$k^{f(\alpha)}$
LONGEST PATH	$O(\frac{n}{\log n})$ [2]	$2^{(\log n)^{1-\epsilon}}$ [32]	no $k^{O(1)}$ [5]	α	no $k^{f(\alpha)}$

Figure 1: Summary of known and new results for the problems considered in this paper. The columns show respectively: the best factor of a known approximation algorithm, the best known lower bound on the approximation ratio of polynomial time approximation algorithms, the best known kernel (or kernel lower bound), the approximation factor of the relevant approximate kernel, and the size of that approximate kernel. In the problem name column, V.C. abbreviates vertex cover.

No Poly Kernel

CONNECTED VERTEX COVER

DISJOINT FACTOR

STEINER TREE

CYCLE PACKING

poly α-kernel

no poly α-kernel

PATH

TREE ISOMORPHISM

Poly Kernel

ODD CYCLE TRANSVERSAL

VERTEX COVER

FEEDBACK VERTEX SET

MIN-ONES-d-SAT

OPEN

MULTIWAY CUT

CHORDAL VERTEX DELETION

DIRECTED FEEDBACK VERTEX SET

How do we make α-approximate kernel

- Safe Reduction Rules
- (I, k) if and only if (I', k').

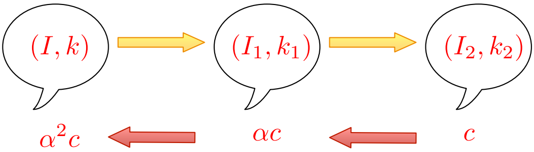
$$(I, k) \iff (I_1, k_1) \iff (I_3, k_3) \cdots \iff (I_\ell, k_\ell)$$

Making α-approximate kernel

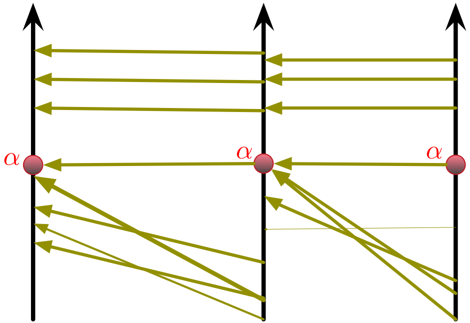
- α-Safe Reduction Rules
- $(I, k) \implies (I', k')$. For every c-approximate solution to (I', k') we get αc approximate solution to original.

Making α-approximate kernel

- α-Safe Reduction Rules
- $(I, k) \implies (I', k')$. For every c-approximate solution to (I', k') we get αc approximate solution to original.

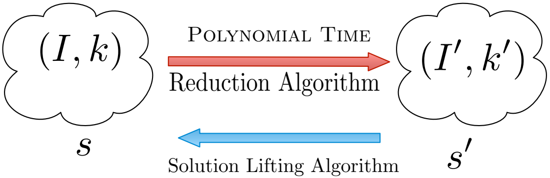


α-Safe Reduction Rule



- 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

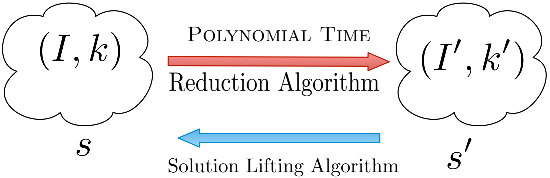


If Π is a minimization problem

then

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

α-Safe Reduction Rule

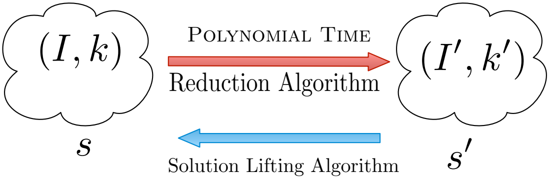


$$\frac{\Pi(I,k,s)}{OPT(I,k)} \leq \max\left\{\frac{\Pi(I',k',s')}{OPT(I',k')}, \alpha\right\}.$$

$$OPT(I', k') \leq OPT(I, k) - \Delta_{\mathsf{forward}}$$

$$\Pi(I, k, s) \leq \Pi(I', k', s') + \Delta_{\mathsf{backward}}$$

α-Safe Reduction Rule



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

$$OPT(I',k') \leq OPT(I,k) - \Delta_{\text{forward}}$$

$$\Pi(I,k,s) \leq \Pi(I',k',s') + \Delta_{\text{backward}}$$

$$\frac{\Pi(I,k,s)}{OPT(I,k)} \leq \frac{\Pi(I',k',s') + \Delta_{\text{backward}}}{OPT(I',k') + \Delta_{\text{forward}}}$$

$$\leq \max\left\{\frac{\Pi(I',k',s')}{OPT(I',k')}, \frac{\Delta_{\text{backward}}}{\Delta_{\text{forward}}}\right\}$$

α-approximate kernel for CVC

Let d be the least integer such that $\frac{d}{d-1} \leqslant \alpha$.
Rule: 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 $k' = k - (D - 1)$.

α-approximate kernel for CVC

Let d be the least integer such that $\frac{d}{d-1} \leqslant \alpha$.
A *false twin* of a vertex v is a vertex u such that $uv \notin E(G)$ and $N(u) = N(v)$.
Rule: If a vertex v has at least $k+1$ false twins, then remove v, i.e output $G' = G - v$ and $k' = k$.

Pick your favourite problem for which has no polynomial kernel and try this approach!

Thank You.