LOSSY KERNELIZATION

PARAMETERIZED COMPLEXITY SUMMER SCHOOL

ALGO 2017

VIENNA

M. S. RAMANUJAN

TU WIEN
LOSSY Kernelization

APPROXIMATION ALGORITHMS

KERNELIZATION
LOSSY Kernelization

Approximation Algorithms

KERNELIZATION

LOSSY

KERNELS
Kernelization

Polynomial Kernel if $|I'| + k' < \text{poly}(k)$
Kernelization

Yes instance $I, k$ $\rightarrow$ polynomial time $\rightarrow$ Yes instance $I', k'$

? $\rightarrow$ a 1.0001-approximate solution $\leftarrow$?
Kernelization

I, k \rightarrow \text{polynomial time} \rightarrow I', k'

Yes instance \rightarrow \text{an optimal solution} \rightarrow \text{poly time} \rightarrow \text{an optimal solution}

Yes instance
Kernelization

$I, k \rightarrow \text{polynomial time} \rightarrow I', k' \leftrightarrow \text{poly time} \leftrightarrow \text{a 1.0001-approximate solution} \\
\text{I,k} \rightarrow \text{a 1.0001-approximate solution} \\
\text{a 1.0001-approximate solution} \rightarrow \text{I',k'}
approximate Kernelization

But if we are allowing a loss while solving $I'$, makes sense to allow a slight loss while lifting a solution back to $I$.
$\alpha$-approximate Kernelization

$\exists (\alpha c)$-approximate solution in polynomial time

[I, k] $\rightarrow$ [polynomial time] $\rightarrow$ [I', k']

$\exists$ a $c$-approximate solution in polynomial time

Polynomial $\alpha$-approximate Kernel if $|I'| + k' < \text{poly}(k)$

[Lokshtanov, Panolan, R., Saurabh, 17]
A problem is FPT if and only if it has a kernel.

A problem has an FPT time $\alpha$-approximation if and only if it has a $\alpha$-approximate kernel.
Previously

A problem is in P if and only if it has a constant size kernel.

A problem has a poly time $\alpha$-approximation if and only if it has a constant size $\alpha$-approximate kernel.

Now
A problem has a poly time $\alpha$-approximation if and only if it has a constant size $\alpha$-approximate kernel.

We are given a poly time $\alpha$-approximate algo.

Want to define this object.

We are given a poly time $\alpha$-approximate algo.

We want to define this object.

We are given a poly time $\alpha$-approximate algo.

We want to define this object.

We are given a poly time $\alpha$-approximate algo.

We want to define this object.

We are given a poly time $\alpha$-approximate algo.

We want to define this object.

We are given a poly time $\alpha$-approximate algo.

We want to define this object.

We are given a poly time $\alpha$-approximate algo.

We want to define this object.

We are given a poly time $\alpha$-approximate algo.

We want to define this object.

We are given a poly time $\alpha$-approximate algo.

We want to define this object.

We are given a poly time $\alpha$-approximate algo.

We want to define this object.

We are given a poly time $\alpha$-approximate algo.

We want to define this object.

We are given a poly time $\alpha$-approximate algo.

We want to define this object.

We are given a poly time $\alpha$-approximate algo.

We want to define this object.

We are given a poly time $\alpha$-approximate algo.

We want to define this object.

We are given a poly time $\alpha$-approximate algo.

We want to define this object.

We are given a poly time $\alpha$-approximate algo.

We want to define this object.

We are given a poly time $\alpha$-approximate algo.

We want to define this object.
A problem has a poly time $\alpha$-approximation if and only if it has a constant size $\alpha$-approximate kernel.
\( \alpha \)-approximate Kernelization

A problem has a poly time \( \alpha \)-approximation if and only if it has a constant size \( \alpha \)-approximate kernel.

Want to design \( \alpha \)-approximate kernel where \( \alpha \) beats best known approx bound and size beats best known kernel bound.
Given a CNF formula, what is the maximum number of clauses which can be satisfied?

Max SAT

APX-hard
[BGLR 93]

no kernel of size poly(n)
[Fortnow Santhanam, 08]

Want to design $\alpha$-approximate kernel where $
\alpha=(1-\varepsilon)$ size is polynomial
Set \( d = \log \left( \frac{2}{\epsilon} \right) \)

if at most \( \epsilon m/2 \) clauses have size at most \( d \)

then a random assignment will leave at most \( \epsilon m/2 + m/2^d = \epsilon m \) clauses unsatisfied
at least \((1-\epsilon)m\) clauses satisfied!

otherwise

\( \epsilon m/2 \leq \# \text{ small clauses} \leq n^{\log(2/\epsilon)} \)

\( m = O(n^{\log(2/\epsilon)}) \)

Max SAT

APX-hard
[BGLR 93]

no kernel of size poly(n)
[FS 08]
Set $d = \log \left( \frac{2}{\varepsilon} \right)$

- at least $(1-\varepsilon)m$ clauses satisfied!
- find a $(1-\varepsilon)$-approximation

OR

- $m = O(n^{\log(2/\varepsilon)})$
- already have an $O(n^c)$ kernel

Max SAT

APX-hard [BGLR 93]

no kernel of size poly(n) [FS 08]
Max SAT

Set $d = \log \left( \frac{2}{\varepsilon} \right)$

- at least $(1-\varepsilon)m$ clauses satisfied!
- $m = O(n^{\log(2/\varepsilon)})$

find a $(1-\varepsilon)$-approximation

OR

already have an $O(n^c)$ kernel

a $(1-\varepsilon)$-approximate kernel of size $O(n^c)$

APX-hard
[BGLR 93]

no kernel of size $\text{poly}(n)$
[FS 08]
Problems parameterized by solution value
Problems parameterized by solution value

What is k in this context? k is a threshold on the size of solutions we want to output.
**Connected Vertex Cover**

Given a graph $G$, find a smallest vertex cover which induces a connected subgraph?

- 2-approximable [AHH 93]
- $2^k$ kernel
- No $(2-\epsilon)$ approximation under UGC [KR 08]
- No polynomial kernel [DLS 09]
Vertex Cover

- $H = \text{vertices of degree at least } k+1$
- $R = \text{vertices with at least one neighbor not in } H.$
- $I = \text{remaining vertices, have all neighbors in } H, \text{ must be independent.}$
Vertex Cover

- \( H = \) vertices of degree at least \( k+1 \)
- \( R = \) vertices with at least one neighbor not in \( H \).
- \( I = \) remaining vertices, have all neighbors in \( H \), must be independent.
CONNECTED Vertex Cover

- \( H \) = vertices of degree at least \( k+1 \)
- \( R \) = vertices with at least one neighbor not in \( H \).
- \( I \) = remaining vertices, have all neighbors in \( H \), must be independent.

Vertices of degree \( > k \) 

\( |I'| + k' < \text{poly}(k) \)
Connected Vertex Cover

(a) a feasible solution
(b) \( \text{OPT}(G) (1+\epsilon) \geq \text{OPT}(G') \)
(c) \( |V(G')| \leq k^{O(1/\epsilon)} \)

**Hint:** Somehow store interesting solutions within \( k^{O(1/\epsilon)} \) vertices.
• Run the 2-approximation algorithm.

• If output > 2k, then say OPT(G) > k and stop.

• Otherwise, OPT(G) ≤ 2k
- **H** = vertices of degree at least 2k+1
- **R** = vertices incident on at least one edge not incident on **H**.
- **I** = remaining vertices, must be independent.
- add a pendant to vertices in **H**.
• $S := H, d = 2/\epsilon$

• As long as there is a vertex $v$ in $I$ seeing $\geq d$ components of $G[S]$, set $S := S \cup \{v\}$, add pendant to $v$.

• Procedure stops after $\leq \epsilon/2 |H| \leq \epsilon \cdot \text{OPT}(G)$ steps

• So, $\text{OPT}(G') \leq \text{OPT}(G) (1+\epsilon)$

• What about size of $G'$?
if $u$ and $v$ have the same neighbouring components in $G[S]$, just delete one.

Finally, we are left with

$$|I| \leq k^{O(d)} = k^{O(1/\epsilon)}$$

Size bound holds.

$(1+\epsilon)$-approximate kernel of size $k^{O(1/\epsilon)}$ for Conn. Vertex Cover.
Open Problem:

Is there a \((1+\epsilon)\)-approx. kernel of size \(f(1/\epsilon) \cdot k^{O(1)}\) for Conn. Vertex Cover.

Efficient PSAKS?

- \((1+\epsilon)\)-approximate kernel of size \(k^{O(1/\epsilon)}\) for Conn. Vertex Cover.
- Polynomial Size Approximate Kernelization Scheme
We just saw a PSAKS for connected vertex cover.
Another example: $H$-hitting set

Given a graph $G$, find a smallest $H$-hitting set which induces a connected subgraph?

$H$ is a fixed finite family of finite graphs. We want to hit every copy of a graph in $H$, which is in $G$.

When $H=\{K_2\}$, then we have the Connected Vertex Cover problem.
The connected H-hitting set problem has a (1+\(\varepsilon\))-approximate kernelization of polynomial size for every 0<\(\varepsilon\)<1.

[European Conference on Evolutionary Computation in Combinatorial Optimization, 2017]
Known kernel for $H$-hitting set

Every (not necessarily connected) $H$-hitting set of $G'$ of size at most $k$ is a (not necessarily connected) $H$-hitting set of $G$.

$|V(G')| = k^{O(d)}$ where $d =$ size of largest graph in $H$
Known kernel for $H$-hitting set

Like for connected vertex cover, vertices outside $G'$ are only needed for connectivity.
Known kernel for H-hitting set

Like for connected vertex cover, vertices outside $G'$ are only needed for connectivity.

But again, which vertices? Even worse, $G \setminus G'$ is not so simple as for vertex cover!

Hint: We only want to preserve approximate connectivity between solution vertices in $V(G')$. 
A Steiner tree for a set of terminals $T$ is a connected subgraph of $G$ spanning the vertices in $T$. 

Digression: Steiner tree approximation
For every error parameter $0 < \epsilon < 1$, there is a $p(1/\epsilon)$ such that any set containing the optimal steiner tree for EVERY $p(1/\epsilon)$-sized subset of $T$, also contains a $(1+\epsilon)$ approximate Steiner Tree for every $R \subseteq T$. 

Borchers and Du, 95
Digression: Steiner tree approximation

For every error parameter $0 < \epsilon < 1$, there is a $p(1/\epsilon)$ such that any set containing the optimal steiner tree for EVERY $p(1/\epsilon)$-sized subset of $T$, also contains a $(1+\epsilon)$ approximate Steiner Tree for every $R \subseteq T$.

Total number of vertices $\leq |T| + q. |T|^{p(1/\epsilon)}$

*Borchers and Du, 95*
Steiner tree approximation in our setting

For every error parameter $0 < \epsilon < 1$, there is a $p(1/\epsilon)$ such that any set containing the optimal steiner tree for EVERY $p(1/\epsilon)$-sized subset of $T$, also contains a $(1+\epsilon)$ approximate Steiner Tree for every $R \subseteq T$.

Borchers and Du, 95

Total number of vertices $\leq |T| + q \cdot |T|^{p(1/\epsilon)}$

But for us, $q = k$ and $|T| = |V(G')| = k^{O(d)}$
Steiner tree approximation in our setting

For every error parameter $0<\epsilon<1$, there is a $p(1/\epsilon)$ such that any set containing the optimal steiner tree for EVERY $p(1/\epsilon)$-sized subset of $T$, also contains a $(1+\epsilon)$ approximate Steiner Tree for every $R \subseteq T$.

Borchers and Du, 95

Total number of vertices $\leq |T|+q.|T|^p(1/\epsilon)$

But for us, $q= k$ and $|T|=|V(G')|=k^{O(d)}$
Steiner Tree

For every error parameter $0 < \epsilon < 1$, there is a $p(1/\epsilon)$-approximate Steiner Tree for every $R \subseteq T$. However, there is no polynomial-sized $(1+\epsilon)$-approximate kernel for Steiner Tree.
Partial Vertex Cover

- No kernel unless FPT=W[1]
- (4/3-\(\varepsilon\))-Appx
- APX-hard

[Marx, 2008]
Partial Vertex Cover

(4/3-\(\epsilon\))-Appx

No kernel unless FPT=W[1]

APX-hard

(1+\(\epsilon\))-approximate kernel of polynomial size

[Marx, 2008]
Cycle Packing

- $k^k \log k$ kernel
- No poly kernel
- Log n-Appx
- No const approx

[Lokshtanov, Panolan, R., Saurabh, 17]
Cycle Packing

- $k^k \log k$ kernel
- No poly kernel
- (1+\(\varepsilon\))-approximate kernel of polynomial size
- \log n–Appx
- no const approx

[Lokshtanov, Panolan, R., Saurabh, 17]
More approximate kernels

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Connected V.C.</td>
<td>2</td>
<td>(2 - (\epsilon)) [41]</td>
<td>(k^{O(1)}) [22]</td>
<td>1 &lt; (\alpha)</td>
<td>(k f(\alpha))</td>
</tr>
<tr>
<td>Cycle Packing</td>
<td>(O(\log n)) [51]</td>
<td>(log (n))^{1-(\epsilon)} [33]</td>
<td>no (k^{O(1)}) [8]</td>
<td>1 &lt; (\alpha)</td>
<td>(k f(\alpha))</td>
</tr>
<tr>
<td>Disjoint Factors</td>
<td>2</td>
<td>no PTAS</td>
<td>(</td>
<td>\Sigma</td>
<td>^{O(1)}) [8]</td>
</tr>
<tr>
<td>Longest Path</td>
<td>(O(\frac{n}{\log n})) [2]</td>
<td>(2^{(\log n)^{1-\epsilon}}) [39]</td>
<td>no (k^{O(1)}) [6]</td>
<td>any (\alpha)</td>
<td>no (k^{O(1)})</td>
</tr>
<tr>
<td>Set Cover/n</td>
<td>(\ln n) [55]</td>
<td>(1 - (\epsilon)) (\ln n) [47]</td>
<td>no (n^{O(1)}) [22]</td>
<td>any (\alpha)</td>
<td>no (n^{O(1)})</td>
</tr>
<tr>
<td>Hitting Set/n</td>
<td>(O(\sqrt{n})) [48]</td>
<td>(2^{(\log n)^{1-\epsilon}}) [48]</td>
<td>no (n^{O(1)}) [22]</td>
<td>any (\alpha)</td>
<td>no (n^{O(1)})</td>
</tr>
<tr>
<td>Vertex Cover</td>
<td>2</td>
<td>(2 - (\epsilon)) [21, 41]</td>
<td>(2k^{d}) [15]</td>
<td>1 &lt; (\alpha &lt; 2)</td>
<td>(2(2 - \alpha)k) [30]</td>
</tr>
<tr>
<td>d-Hitting Set</td>
<td>(d) [55]</td>
<td>(d - \epsilon) [20, 41]</td>
<td>(O(k^{d-1})) [1]</td>
<td>1 &lt; (\alpha &lt; d)</td>
<td>(O((k \cdot \frac{d-\alpha}{\alpha-1})^{d-1})) [30]</td>
</tr>
<tr>
<td>Steiner Tree</td>
<td>1.39 [11]</td>
<td>no PTAS [12]</td>
<td>no (k^{O(1)}) [22]</td>
<td>1 &lt; (\alpha)</td>
<td>(k f(\alpha))</td>
</tr>
<tr>
<td>OLA/v.c.</td>
<td>(O(\sqrt{\log n \log \log n})) [28]</td>
<td>no PTAS [3]</td>
<td>(f(k)) [43]</td>
<td>1 &lt; (\alpha &lt; 2)</td>
<td>(f(\alpha)2^k)</td>
</tr>
<tr>
<td>Partial V.C.</td>
<td>((\frac{4}{3} - \epsilon)) [27]</td>
<td>no PTAS [49]</td>
<td>no (f(k)) [35]</td>
<td>1 &lt; (\alpha)</td>
<td>(f(\alpha)k^3)</td>
</tr>
</tbody>
</table>

Lot of these problems have played a critical role in the development of kernel lower bound theory
Longest Path
Given a graph $G$, what is the length of a longest path in $G$?
Longest Path

- No poly kernel
- No c-approximate poly kernel unless NP in coNP/Poly
- no const approx
Longest Path

No poly kernel

no const approx

No c-approximate poly kernel unless NP in coNP/Poly
Lower Bounds

Ruling out c-approximate polynomial kernels

Ruling out polynomial kernels

Cross-compositions

Ruling out c-approximations

Gap-creating reductions

Gap-creating Cross Compositions
Lower Bounds

Gap-creating Cross Compositions

If some \((x_i, k)\) is a yes-instance
then \(\text{OPT}(y, k')\) is LARGE
Otherwise \(\text{OPT}(y, k')\) is SMALL

\(k' = \text{poly}(\max (|x_i|) + \log t)\)

MAXIMIZATION PROBLEM

real number \(r\)

Poly(input)
Lower Bounds

````α'''''' Gap-creating Cross Compositions

\[(x_1, k)\] \[(x_2, k)\] \[(x_3, k)\] \[\ldots\] \[(x_t, k)\]

\[k' = \text{poly}(\max (|x_i|) + \log t)\]

If some \((x_i, k)\) is a yes-instance then \(\text{OPT}(y, k') \geq r\)

Otherwise \(\text{OPT}(y, k') < r/\alpha\)

real number \(r\) \((y, k')\)
Lower Bounds

\[ \alpha "]

Gap-creating Cross Compositions

Problem P

\[ \alpha \]-Gap-creating Cross Composition + \[ \alpha \]-approximate polynomial compression
Lower Bounds

$\alpha$-gap Long Path cross composes into Longest Path

Longest Path has no $\alpha$-approximate polynomial kernel unless NP in coNP/poly.
More Lower Bounds

Set Cover (universe size) has no \( \alpha \)-approximate polynomial kernel unless NP in coNP/poly.
α-approximate Poly. Param. Transformations

Reductions to rule out approximate polynomial kernels

Problem P

Poly time reduction

α-APPT

Problem Q

αc-approx for P ← c-approx for Q


α-approximate Poly. Param. Transformations

Reductions to rule out approximate polynomial kernels

Problem P → \( \alpha \)-APPT → compress(Q)

\( \alpha \)-approx for P \( \sim \) c-approx for Q

Poly time reduction

\( \beta/\alpha \)-approx. poly compression
\( \alpha \)-approximate Poly. Param. Transformations

Reductions to rule out approximate polynomial kernels

Problem P

\( \beta \)-approximate polynomial compression

compress(Q)

no \( \beta \)-approximate polynomial compression for \( P + \alpha \)-APPT from \( P \) to \( Q \) \( \rightarrow \)

no \( \beta/\alpha \)-approx. poly compression for \( Q \)
Other questions to attack using lossy kernels
Other questions to attack using lossy kernels
Other questions to attack using lossy kernels

No reason why the study of approximate kernels should be restricted to problems without polynomial kernels
approx factor vs kernel size

Dominating Set on \( d \)-degenerate graphs

- A vertex set \( S \) such that every vertex in \( V(G) \setminus S \) is adjacent to a vertex in \( S \).
- A graph where every subgraph has a vertex of degree at most \( d \).

There is a \( d^2 \)-approximation \([\text{Jones et al.}]\).

There is a kernel of size \( k^{O(d^2)} \) \([\text{Philip et al.}]\).

Cannot have a kernel of size \( k^{o(d^2)} \) \([\text{Cygan et al.}]\).
Dominating Set on $d$-degenerate graphs

- A vertex set $S$ such that every vertex in $V(G) \setminus S$ is adjacent to a vertex in $S$.
- A graph where every subgraph has a vertex of degree at most $d$.

There is a $d^2$-approximate kernel of size $O(1)$.

There is a 1-approximate kernel of size $k^{O(d^2)}$. 

approx factor vs kernel size
Dominating Set on $d$-degenerate graphs

- A vertex set $S$ such that every vertex in $V(G) \setminus S$ is adjacent to a vertex in $S$.
- A graph where every subgraph has a vertex of degree at most $d$.

There is a $d^2$-approximate kernel of size $O(1)$.
There is a 1-approximate kernel of size $k^{O(d^2)}$.

Is there a curve interpolating between these two extremes?
Dominating Set on $d$-degenerate graphs

- A vertex set $S$ such that every vertex in $V(G) \setminus S$ is adjacent to a vertex in $S$.
- A graph where every subgraph has a vertex of degree at most $d$.

For every $\rho$ in $\{1, \ldots, d\}$ there is a $(d/\rho)$-approximate kernel of size $k^{O(\rho d)}$.

[Refs: Eiben, Hermelin, R. 17]
approx factor vs kernel size

Open Problem:
    What about d-hitting set?

There is a d-approximate kernel of size $O(1)$
There is a 1-approximate kernel of size $k^{O(d)}$

Is there a curve interpolating between these two extremes?
Uniform vs non-uniform kernels

Treewidth-$t$ deletion problem has a kernel of size $k^{f(t)}$
[Fomin, Misra, Lokshtanov, Saurabh, 12]

Treewidth-$t$ deletion problem does not have a kernel of size $f(t) \cdot k^{O(1)}$ unless NP in coNP/poly.
[Giannopolou, Jansen, Lokshtanov, Saurabh, 15]

Treewidth-$t$ deletion problem has a $(1+\varepsilon)$-approximate kernel of size $f(t) \cdot k^3$
[Koutecký, Lokshtanov, Misra, Saurabh, Sharma, Zehavi 17]
Take home message

• For many problems, allowing a small loss in accuracy, gives a dramatic improvement in kernel size.

• Interesting questions even for problems with poly kernels!

• Does not work for all problems! There is a lower bound machinery.

• If you like kernelization and/or approximation, you’ll probably like their as well!
Open Problems

Many many many open problems.

Full version of main paper on arxiv has a list of problems.

What about Directed Feedback Vertex Set?
Thank you for your attention!
Thank you for your attention!
Thank you for your attention!
Thank you for your attention!