Kernelization Lower Bounds: A Brief History

G Philip

Max Planck Institute for Informatics, Saarbrücken, Germany

New Developments in Exact Algorithms and Lower Bounds.
Pre-FSTTCS 2014 Workshop, IIT Delhi
December 14, 2014

Parameterized Complexity

A brief review

- One way of coping with NP-hard problems

Example (VERTEX COVER, standard parameterization)

- Input:
  - A graph $G = (V, E)$
  - A positive integer k
- Question: Is there a set $S \subseteq V$ of at most k vertices (a *vertex cover* of G) such that every edge in G has at least one vertex of S as an end-point?
- "Standard" parameter: The number k

Notions of Tractability

Definition (Fixed-parameter tractability)

A parameterized problem is *fixed-parameter tractable* (FPT) if there is an algorithm which solves instances (x, k) of the problem in time $f(k) \cdot |x|^c$ where

- $f()$ is a computable function of k alone;
- c is a constant, independent of k and $|x|$.

Example (Vertex Cover is FPT)

- A simple branching algorithm which runs in $\mathcal{O}(2^k \cdot |G|)$ time.

Notions of Tractability

Fixed-parameter tractability

Problem	$\mathbf{f(k)}$
Vertex Cover	1.2738^k
Feedback Vertex Set	3.619^k
d-Hitting Set	$(d - 1 + \varepsilon)^k$
k-Path	4^k
Connected Vertex Cover	2^k
Steiner Tree	2^k
Directed Feedback Vertex Set	$4^k \cdot k!$
⋮	⋮

- From the Table of FPT Races at `http://fpt.wikidot.com/fpt-races`.

Notions of Tractability

Fixed-parameter tractability

- The corresponding notion of ***in***tractability: W-hardness.
- If a parameterized problem is W-complete, then it is unlikely to be FPT
  - Because they "must all hang together, or they shall all hang separately"
  - Just like NP-completeness
  - Lots of examples of W-hard problems
  - Standard parameterizations of INDEPENDENT SET (so also CLIQUE), DOMINATING SET, ...

Definition (Kernelization, Kernel, Kernel size)

A *kernelization algorithm* for a parameterized problem is an algorithm which, given an input (x, k) of the problem,

- Runs in time *polynomial* in $|x| + k$;
- Outputs an instance (x', k') of the problem where:
  - (x', k') is a **Yes** instance iff (x, k) is a **Yes** instance, and,
  - $|x'|, k' \leq g(k)$ for some computable function $g()$

- (x', k') is called a *kernel*
- $g(k)$ is the *size* of the kernel

Example (The "Buss" kernel for Vertex Cover)

- Observation: If a vertex is not in a vertex cover, then *all* its neighbours *must* be in that vertex cover.
- Implication: Every vertex of degree more than k must be in *any* vertex cover of size at most k.
- Algorithm:
  - Pick all vertices of degree more than k
  - If these are already more than k, then return **No**
  - Now: if there are more than k^2 edges left, then return **No**
  - Return the remaining graph: a kernel with $\mathcal{O}(k^2)$ vertices and edges

Notions of Tractability

Kernelization

Problem	**f(k)**	**Size of the smallest known kernel**
Vertex Cover	1.2738^k	$\mathcal{O}(k^2)$
Feedback Vertex Set	3.619^k	$\mathcal{O}(k^2)$
d-Hitting Set	$(d-1+\varepsilon)^k$	$\mathcal{O}(k^d)$
k-Path	4^k	4^k
Connected Vertex Cover	2^k	2^k
Steiner Tree	2^k	2^k
Directed Feedback Vertex Set	$4^k \cdot k!$	$4^k \cdot k!$

Notions of Tractability

The "first theorem" of Parameterized Complexity

Theorem

A parameterized problem is fixed-parameter tractable if and only if it has a kernel.

Remark

The proof of the more interesting direction shows that if a problem can be solved in $f(k) \cdot n^c$ time then it has a kernel of size $f(k)$.

Notions of Tractability

Kernelization lower bounds I

- What is a corresponding notion of ***in***tractability?
- The theorem rules out kernels of *any size* for W-hard problems*
- What about problems which *are* FPT?
  - The (proof of the) theorem gives kernels of size $f(k)$
  - $f(k)$ is exponential in k for NP-hard problems†
  - We have polynomial-size kernels for many FPT problems
  - Which FPT problems do *not* have polynomial kernels?
  - How do we go about proving such lower bounds?

*Under widely believed assumptions.

†For sensible parameters k, and if solving NP-hard problems takes exponential time.

Notions of Tractability

Kernelization lower bounds II

- What about problems which *do* have polynomial-size kernels?
- Kernel sizes tend to decrease with passing years
- Example: FEEDBACK VERTEX SET
  - First polynomial-size kernel: $\mathcal{O}(k^{11})$ (Burrage et al., 2006)
  - Improved to: $\mathcal{O}(k^3)$ (Bodlaender, 2007)
  - Current best: $\mathcal{O}(k^2)$ (Thomassé, 2009)
- How far can this go on?
- When do we know to stop?
- How do we prove lower bounds on the polynomial *degrees* of kernel sizes?

A (somewhat) different look at kernelization

- Given an instance of a (classical) decision problem:
  - How *small* can we make it in polynomial time, *without* losing the **Yes**/**No** answer?
- If the problem is in P, then we can reduce it all the way to 1 bit
- If the problem is NP-hard, then we cannot[a] reduce its size
  - Even by *one bit*, *without* losing the **Yes**/**No** answer.
  - (Otherwise, we could repeat the procedure and solve the problem in PTIME.)

---

[a] Unless P=NP.

A (somewhat) different look at kernelization

- What is a "correct" question to ask about the polynomial-time "compressibility" of NP-hard problems?
- The PC view: ask how small we can make an instance *in terms of the parameter*, in polynomial time
- When we ask for kernels and kernel-size lower bounds, we are asking the question "What can we (not) do in polynomial time?"
- For a more refined notion of "do" which is relevant for NP-hard problems

Compressing CLIQUE

A non-standard parameterization

Definition (CLIQUE parameterized by number of vertices)

- Input:
  - A graph $G = (V, E)$ on n vertices
  - A positive integer k
- Question: Is there a set $S \subseteq V$ of *at least k* vertices (a *clique*) in G such that there is an edge in G between every pair of vertices in S?
- Parameter: The number n

- What is the smallest kernel for this parameterization of CLIQUE?

Compressing CLIQUE

A non-standard parameterization

- How much can we compress CLIQUE in polynomial time w/o losing the **Yes/No** answer?
  - Recall: the size of the kernel is measured in terms of the parameter, here n.
- A kernel of size $\mathcal{O}(n^2)$ is easy:
  - Encode G as its adjacency matrix: $\mathcal{O}(n^2)$ bits
  - Encode k in binary: $\mathcal{O}(\log n)$ bits
- Is this trivial encoding for CLIQUE the best we can do in polynomial time?
  - The size of the encoding is measured in terms of n
    - n is *not* the size of the input instance here!
  - An encoding into, say, $n^{\frac{3}{2}}$ bits does not directly imply that P=NP
- A question about kernel lower bounds!

Summarizing . . .

- Kernelization is polynomial-time reduction in instance size
- Sizes are measured in terms of a parameter
- A parameterized problem has a kernel (of some size) iff it is FPT
- Interesting questions:
  - How do we separate FPT problems which have *polynomial-size* kernels, from those which don't?
  - How do we prove lower-bounds on the polynomial *degree* of problem kernels?
- The latter question is interesting from a purely classical pov as well (e.g: CLIQUE.)

This Talk

Outline

- ~~Introduction~~
- Ruling out polynomial-size kernels
  - For problems which *do* have exponential-size kernels
    - AKA problems which have FPT algorithms
  - Based on Fortnow and Santhanam (STOC 2008), Bodlaender et al (ICALP 2008).
- Lower-bounding the degrees of polynomial-size kernels
  - Can we have smaller-than-$\mathcal{O}(k^2)$ kernels for VERTEX COVER or FEEDBACK VERTEX SET ...
  - Or compress CLIQUE to less than n^2 bits in polynomial time?
  - Based on Dell and van Melkebeek (STOC 2010).

Ruling Out Polynomial Kernels

Based on . . .

- *On problems without polynomial kernels*
  - Bodlaender, Downey, Fellows and Hermelin,
  - ICALP 2008, JCSS 2009
- *Infeasibility of instance compression and succinct PCPs for NP*
  - Fortnow and Santhanam
  - STOC 2008, JCSS 2011

Composition algorithms

- Simple criterion for ruling out polynomial kernels
  - Simple to understand
  - Not always easy to *apply*!

OR-Composition Algorithms

For parameterized problems

Definition

An OR-composition algorithm for a parameterized problem L is an algorithm that:

- Takes as input a list of instances $((x_1, k), (x_2, k), \ldots, (x_t, k))$ for any integer t;
- Runs in time polynomial in $\sum_i (|x_i| + k)$;
- And outputs an instance (y, k') such that
  1. $(y, k') \in L$ if and only if *at least one* $(x_i, k) \in L$
  2. k' is polynomial in k.

OR-Composition Algorithms

For parameterized problems

Definition

An OR-composition algorithm for a parameterized problem L is an algorithm that:

- Takes as input a list of instances $((x_1, k), (x_2, k), \ldots, (x_t, k))$ for any integer t;
- Runs in time polynomial in $\sum_i (|x_i| + k)$;
- And outputs an instance (y, k') such that
  1. $(y, k') \in L$ if and only if *at least one* $(x_i, k) \in L$
  2. k' is polynomial in k.

Example (OR-composition)

- k-PATH: Does graph G have a simple path of length at least k?

Polynomial kernel lower bounds

Theorem (Bodlaender et al., Fortnow and Santhanam)

*Let L be a parameterized problem whose underlying classical problem is NP-complete. Then **at most one** of the following is true:*

- *L has an OR-composition;*
- *L has a polynomial-size kernel,*

unless $\mathsf{coNP} \subseteq \mathsf{NP}/\mathsf{poly}$.

Remark

The condition $\mathsf{coNP} \subseteq \mathsf{NP}/\mathsf{poly}$ is considered unlikely, because it implies a collapse in the Polynomial Hierarchy.

Polynomial kernel lower bounds

Theorem (Bodlaender et al., Fortnow and Santhanam)

*Let L be a parameterized problem whose underlying classical problem is NP-complete. Then **at most one** of the following is true:*

- *L has an OR-composition;*
- *L has a polynomial-size kernel,*

unless $\mathsf{coNP} \subseteq \mathsf{NP/poly}$.

Remark

The condition $\mathsf{coNP} \subseteq \mathsf{NP/poly}$ is considered unlikely, because it implies a collapse in the Polynomial Hierarchy.

Corollary

k-Path does not have a polynomial-size kernel, unless $\mathsf{coNP} \subseteq \mathsf{NP/poly}$.

Some consequences of the Theorem

Problems with no polynomial kernels unless coNP ⊆ NP/poly

- Essentially every NP-complete problem which asks for a "subgraph of some kind": k-PATH, k-CYCLE, k-EXACT CYCLE, k-MINOR ORDER TEST, k-PLANAR SUBGRAPH TEST, k-BOUNDED TREEWIDTH SUBGRAPH TEST, . . .
- Many NP-complete problems parameterized by the *treewidth* of the input graph: w-VERTEX COVER, w-INDEPENDENT SET, w-CLIQUE, w-DOMINATING SET
- Many more problems, using clever composition techniques and reductions. E.g: k-DISJOINT CYCLES, k-DISJOINT PATHS (Bodlaender, Thomassé, Yeo, ESA 2009), CONNECTED VERTEX COVER, STEINER TREE (Dom, Lokshtanov, Saurabh, ICALP 2009)
- Lots of problems by now!

Revisiting the table ...

Problem	f(k)	Kernel size
Vertex Cover	1.2738^k	$\mathcal{O}(k^2)$
Feedback Vertex Set	3.619^k	$\mathcal{O}(k^2)$
d-Hitting Set	$(d-1+\varepsilon)^k$	$\mathcal{O}(k^d)$
k-Path	4^k	No $k^{\mathcal{O}(1)}$
Connected Vertex Cover	2^k	No $k^{\mathcal{O}(1)}$
Steiner Tree	2^k	No $k^{\mathcal{O}(1)}$
Directed Feedback Vertex Set	$4^k \cdot k!$	$4^k \cdot k!$

AND-Composition

Replace "at least one instance" with "all instances"

Theorem (Bodlaender et al., ICALP 2008)

k-TREEWIDTH *(and many other problems) does not have polynomial-size kernels unless NP-complete problems can have AND-distillation algorithms.*

- Bodlaender et al. thought it unlikely that NP-complete problems have AND-distillation algorithms
- They could not connect this to any complexity-theoretic assumption.

AND-Composition

Replace “at least one instance” with “all instances”

Theorem (Bodlaender et al., ICALP 2008)

k-TREEWIDTH *(and many other problems) does not have polynomial-size kernels unless NP-complete problems can have AND-distillation algorithms.*

- Bodlaender et al. thought it unlikely that NP-complete problems have AND-distillation algorithms
- They could not connect this to any complexity-theoretic assumption.

Theorem (Drucker, FOCS 2012)

If NP-complete problems have AND-distillation algorithms, then $\mathsf{coNP} \subseteq \mathsf{NP/poly}$.

Lower-Bounding the Degrees of Polynomial Kernels

Based on . . .

- *Satisfiability allows no nontrivial sparsification unless the polynomial-time hierarchy collapses*
  - Dell and van Melkebeek
  - STOC 2010, JACM 2014

An Oracle Communication Protocol

- Two players, Alice and Bob
  - Alice is polynomially bounded, Bob has unbounded computational power
- Together, they want to decide if a string x belongs to a specified language L
  - In the beginning, Alice holds the string x
  - In the end, Alice should know if $x \in L$
  - They can communicate with each other to achieve this
  - The *cost* of this protocol is the number of bits sent *from* Alice *to* Bob
    - The bits sent from Bob to Alice do not count in the cost

An Oracle Communication Protocol

- Two players, Alice and Bob
  - Alice is polynomially bounded, Bob has unbounded computational power
- Together, they want to decide if a string x belongs to a specified language L
  - In the beginning, Alice holds the string x
  - In the end, Alice should know if $x \in L$
  - They can communicate with each other to achieve this
  - The *cost* of this protocol is the number of bits sent *from* Alice *to* Bob
    - The bits sent from Bob to Alice do not count in the cost

- Again: "What can we (not) do in polynomial time?"
- For yet another notion of "do"

An Oracle Communication Protocol

- Two players, Alice and Bob
  - Alice is polynomially bounded, Bob has unbounded computational power
- Together, they want to decide if a string x belongs to a specified language L
  - In the beginning, Alice holds the string x
  - In the end, Alice should know if $x \in L$
  - They can communicate with each other to achieve this
  - The *cost* of this protocol is the number of bits sent *from* Alice *to* Bob
    - The bits sent from Bob to Alice do not count in the cost

- A generalization of kernelization
- E.g: VERTEX COVER has a protocol of cost $\mathcal{O}(k^2)$
  1. Alice computes a kernel of size $\mathcal{O}(k^2)$
  2. She sends the kernel to Bob
  3. Bob solves the instance and sends **Yes** or **No** back to Alice
  4. Total cost: $\mathcal{O}(k^2)$

An Oracle Communication Protocol

- Two players, Alice and Bob
  - Alice is polynomially bounded, Bob has unbounded computational power
- Together, they want to decide if a string x belongs to a specified language L
  - In the beginning, Alice holds the string x
  - In the end, Alice should know if $x \in L$
  - They can communicate with each other to achieve this
  - The *cost* of this protocol is the number of bits sent *from* Alice *to* Bob
    - The bits sent from Bob to Alice do not count in the cost

Theorem(Dell and van Melkebeek)

VERTEX COVER admits no protocol of cost $\mathcal{O}(k^{2-\varepsilon})$ where k is the standard parameter, unless coNP $\subseteq$ NP/poly.

Some More Lower Bounds

All these carry over directly to the standard parameterizations

Theorem

VERTEX COVER admits no protocol of cost $\mathcal{O}(n^{2-\varepsilon})$ where n is the number of vertices, unless coNP $\subseteq$ NP/poly. So also for CLIQUE.

Theorem

More generally: for any $d \geq 2$, d-Hitting Set over a universe of size n admits no protocol of cost $\mathcal{O}(n^{d-\varepsilon})$, unless coNP $\subseteq$ NP/poly.

Theorem

Let Π be a nontrivial graph property that is inherited by subgraphs. There is no protocol of cost $\mathcal{O}(k^{2-\varepsilon})$ for deciding if a graph satisfying Π can be obtained from a given graph by deleting at most k vertices, unless coNP $\subseteq$ NP/poly.

Corollary

FEEDBACK VERTEX SET has no kernel of size $\mathcal{O}(k^{2-\varepsilon})$ unless ...

The table, one final time

Problem	$\mathbf{f(k)}$	Kernel size
Vertex Cover	1.2738^k	$\mathcal{O}(k^2)$; No $\mathcal{O}(k^{2-\varepsilon})$
Feedback Vertex Set	3.619^k	$\mathcal{O}(k^2)$; No $\mathcal{O}(k^{2-\varepsilon})$
d-Hitting Set	$(d-1+\varepsilon)^k$	$\mathcal{O}(k^d)$; No $\mathcal{O}(k^{d-\varepsilon})$
k-Path	4^k	No $k^{\mathcal{O}(1)}$
Connected Vertex Cover	2^k	No $k^{\mathcal{O}(1)}$
Steiner Tree	2^k	No $k^{\mathcal{O}(1)}$
Directed Feedback Vertex Set	$4^k \cdot k!$	$4^k \cdot k!$

A closer look at the lower bounds

- VERTEX COVER: Kernels with $\mathcal{O}(k^2)$ edges, no kernel with $\mathcal{O}(k^{2-\varepsilon})$ edges
- What about the number of *vertices* in a kernel?
  - The relaxed VERTEX COVER LP has the half-integrality property
    - Can find an optimal $\{0, \frac{1}{2}, 1\}$-solution in PTIME
  - Theorem (Nemhauser and Trotter, 1975): There is a smallest vertex cover which contains all the 1s and none of the 0s
  - All the $\frac{1}{2}$s together induce a kernel with $\leq 2k$ vertices

A closer look at the lower bounds

- VERTEX COVER: Kernels with $\mathcal{O}(k^2)$ edges, no kernel with $\mathcal{O}(k^{2-\varepsilon})$ edges
- What about the number of *vertices* in a kernel?
  - The relaxed VERTEX COVER LP has the half-integrality property
    - Can find an optimal $\{0, \frac{1}{2}, 1\}$-solution in PTIME
  - Theorem (Nemhauser and Trotter, 1975): There is a smallest vertex cover which contains all the 1s and none of the 0s
  - All the $\frac{1}{2}$s together induce a kernel with $\leq 2k$ vertices
- Upper bound on #vertices in a kernel: $\mathcal{O}(k)$

A closer look at the lower bounds

- Vertex Cover: Kernels with $\mathcal{O}(k^2)$ edges, no kernel with $\mathcal{O}(k^{2-\varepsilon})$ edges
- What about the number of *vertices* in a kernel?
  - The relaxed Vertex Cover LP has the half-integrality property
    - Can find an optimal $\{0, \frac{1}{2}, 1\}$-solution in PTIME
  - Theorem (Nemhauser and Trotter, 1975): There is a smallest vertex cover which contains all the 1s and none of the 0s
  - All the $\frac{1}{2}$s together induce a kernel with $\leq 2k$ vertices
- Upper bound on #vertices in a kernel: $\mathcal{O}(k)$
- *Lower* bound on #vertices in a kernel: $\Omega(k)$
  - Follows directly from the size lower bound
  - n-vertex graphs can be encoded with $\mathcal{O}(n^2)$ bits
  - E.g: An $\mathcal{O}(k^{\frac{3}{4}})$-vertex kernel would have total size $\mathcal{O}(k^{\frac{3}{2}}) = \mathcal{O}(k^{2-\frac{1}{2}})$ bits, contradiction.

A closer look at the lower bounds

- VERTEX COVER:
  - Kernels with $\mathcal{O}(k^2)$ edges, no kernel with $\mathcal{O}(k^{2-\varepsilon})$ edges
  - Kernels with $\mathcal{O}(k)$ vertices, no kernel with $\mathcal{O}(k^{1-\varepsilon})$ edges
- FEEDBACK VERTEX SET:
  - Kernels with $\mathcal{O}(k^2)$ edges, no kernel with $\mathcal{O}(k^{2-\varepsilon})$ edges
  - Current upper bound on #vertices: $\mathcal{O}(k^2)$
  - Dell and van Melkebeek only rule out kernels with $\mathcal{O}(k^{1-\varepsilon})$ vertices
  - Gap!

A closer look at the lower bounds

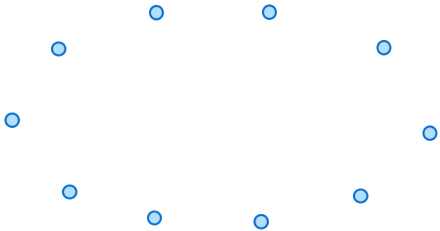
- Vertex Cover:
  - Kernels with $\mathcal{O}(k^2)$ edges, no kernel with $\mathcal{O}(k^{2-\varepsilon})$ edges
  - Kernels with $\mathcal{O}(k)$ vertices, no kernel with $\mathcal{O}(k^{1-\varepsilon})$ edges
- Feedback Vertex Set:
  - Kernels with $\mathcal{O}(k^2)$ edges, no kernel with $\mathcal{O}(k^{2-\varepsilon})$ edges
  - Current upper bound on #vertices: $\mathcal{O}(k^2)$
  - Dell and van Melkebeek only rule out kernels with $\mathcal{O}(k^{1-\varepsilon})$ vertices
  - Gap!
- d-Hitting Set:
  - Best known kernels have $\mathcal{O}(k^d)$ sets over a universe of size $\mathcal{O}(k^{d-1})$
  - Dell and van Melkebeek rule out kernels with $\mathcal{O}(k^{d-\varepsilon})$ sets or a universe of size $\mathcal{O}(k^{1-\varepsilon})$
  - Gap!

A tight non-trivial “structural” kernel lower bound

- For a variant of Hitting Set
- The first result of this kind
- An application of the full power of the protocol
- *Point Line Cover: The Easy Kernel is Essentially Tight*
  - Stefan Kratsch, G. Philip, and Saurabh Ray, SODA 2014

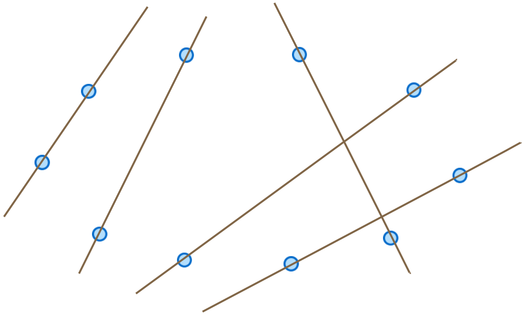
The Point-Line Cover problem

- Input:
  - A set $\mathcal{P} = \{p_1, p_2, \ldots, p_n\}$ of n points in the plane
    - Each point is a pair of rational coordinates: $p_i = (x_i, y_i)$
  - A positive integer k
- Question: Is there a set $\mathcal{L}$ of at most k lines in the plane which together *cover* all points in $\mathcal{P}$?
  - Each point in the set $\mathcal{P}$ must lie on at least one of the lines in $\mathcal{L}$.



The Point-Line Cover problem

- Input:
  - A set $\mathcal{P} = \{p_1, p_2, \ldots, p_n\}$ of n points in the plane
    - Each point is a pair of rational coordinates: $p_i = (x_i, y_i)$
  - A positive integer k
- Question: Is there a set $\mathcal{L}$ of at most k lines in the plane which together *cover* all points in $\mathcal{P}$?
  - Each point in the set $\mathcal{P}$ must lie on at least one of the lines in $\mathcal{L}$.



The Point-Line Cover problem

- NP-hard (Megiddo and Tamir, 1982)
- Standard parameter: k
- Kernel with $\leq k^2$ *points*
  - Langerman and Morin, 2005
  - Uses the "Buss" idea, like for VERTEX COVER
- Open: Is there a kernel with $o(k^2)$ points?

Our Result

$\varepsilon > 0$ is any positive constant

Theorem

The Point-Line Cover problem does not have a kernel with $\mathcal{O}(k^{2-\varepsilon})$ *points* unless coNP $\subseteq$ NP/poly.

Our Result

$\varepsilon > 0$ is any positive constant

Theorem

The Point-Line Cover problem does not have a kernel with $\mathcal{O}(k^{2-\varepsilon})$ *points* unless coNP $\subseteq$ NP/poly.

- This does *not* rule out kernels with, say, $\mathcal{O}(\frac{k^2}{\log k}) = o(k^2)$ points
- We use $\Omega(k^2)$ to denote a bound like in the theorem.

Tight bound for #points in Point-Line Cover kernels

A first attempt

- We have: $\mathcal{O}(k^2)$ upper bound on #*points*
- We want: $\Omega(k^2)$ lower bound on #*points*
- How?

Tight bound for #points in Point-Line Cover kernels

A first attempt

- We have: $\mathcal{O}(k^2)$ upper bound on *#points*
- We want: $\Omega(k^2)$ lower bound on *#points*
- How?
- We derive: $\Omega(k^2)$ lower bound on *total size*
  - The $\Omega(k^2)$ lower bound on VERTEX COVER kernel size
  - Reduction from VERTEX COVER to Point-Line Cover
    - $k \to 2k$

Tight bound for #points in Point-Line Cover kernels

A first attempt

- We have: $\Omega(k^2)$ lower bound on *total size*
- We want: An $\mathcal{O}(n \cdot polylog(n))$-bit polynomial-time encoding of Point-Line Cover instances with n points

Tight bound for #points in Point-Line Cover kernels

A first attempt

- We have: $\Omega(k^2)$ lower bound on *total size*
- We want: An $\mathcal{O}(n \cdot polylog(n))$-bit polynomial-time encoding of Point-Line Cover instances with n points
- The best known such encoding has $\mathcal{O}(n^2)$ bits
- This gives: $\Omega(k)$ lower bound on *#points* in a kernel
  - E.g: An $\mathcal{O}(k^{3/4})$-point kernel implies a kernel of *total size* $\mathcal{O}(k^{3/2})$
  - Contradicting the $\Omega(k^2)$ lower bound on kernel size
  - Doesn't rule out kernels with, say, $\mathcal{O}(k^{\frac{3}{2}})$ points
  - Such a kernel has total size $\mathcal{O}(k^3)$ bits, contradicting nothing

Tight bound for #points in Point-Line Cover kernels

A first attempt

- The best-known encoding gives : $\Omega(k)$ lower bound on #points
- One way to improve this to $\Omega(k^2)$: Find an $\mathcal{O}(n \log n)$-bit polynomial-time encoding for n-point instances
  - Open since the very first SOCG (1985)
  - It is *known* that there exists such an encoding
  - The hard (and unknown) part is to find it in polynomial time
- We achieve this *without* finding a better encoding
- Using the Oracle Communication Protocol

An Outline of the Proof

- Recall: A Point-Line Cover instance is $(\mathcal{P}, k)$; $\mathcal{P}$ is a set of n points.
- The proof has two main ingredients:
  1. A lower bound of $\Omega(k^2)$ on the cost of a protocol for Point-Line Cover
  2. An *upper* bound of $\mathcal{O}(n \log n)$ on the cost of a protocol for Point-Line Cover
- Together, these give us a lower bound of $\Omega(k^2)$ on the *number of points* in a kernel

An Outline of the Proof

- Recall: A Point-Line Cover instance is $(\mathcal{P}, k)$; $\mathcal{P}$ is a set of n points.
- The proof has two main ingredients:
  1. A lower bound of $\Omega(k^2)$ on the cost of a protocol for Point-Line Cover
  2. An *upper* bound of $\mathcal{O}(n \log n)$ on the cost of a protocol for Point-Line Cover
- Together, these give us a lower bound of $\Omega(k^2)$ on the *number of points* in a kernel
- Suppose there was a kernel for Point-Line Cover with $k^{2-\varepsilon}$ *points*
  - Alice is given an instance $(\mathcal{P}, k)$; $|\mathcal{P}| = n$ of Point-Line Cover
  - She computes kernel $(\mathcal{P}', k')$ with $n' = |\mathcal{P}'| = k^{2-\varepsilon}$ points
  - Alice and Bob use the second ingredient to decide $(\mathcal{P}', k')$
  - Cost: $\mathcal{O}(n' \log n') = \mathcal{O}(k^{2-\varepsilon} \log(k^{2-\varepsilon})) = \mathcal{O}(k^{2-\varepsilon} \log k) = \mathcal{O}(k^{2-\varepsilon'})$
  - This contradicts the cost lower bound

The Lower Bound

A brief look

Theorem

The Point-Line Cover problem does not admit an oracle communication protocol of cost $\mathcal{O}(k^{2-\varepsilon})$ unless $\mathsf{coNP} \subseteq \mathsf{NP/poly}$.

- Outline of the proof:
  - Polynomial-time, parameter-preserving reduction from VERTEX COVER to Point-Line Cover
    - (G, k) goes to $(\mathcal{P}, 2k)$
  - The theorem now follows from the $\Omega(k^{2-\varepsilon})$ lower bound on the cost of VERTEX COVER protocols

The Upper Bound

A closer look

Theorem

There is an oracle communication protocol which can solve Point-Line Cover instances with *n points* at a cost of $\mathcal{O}(n \log n)$.

The Upper Bound

A first attempt at such a protocol

- Given an instance $(\mathcal{P}, k)$; $|\mathcal{P}| = n$ of Point-Line Cover
  - Alice computes an encoding X of $\mathcal{P}$, where X has $\mathcal{O}(n \log n)$ bits
  - She then sends X over to Bob
    - Cost: $\mathcal{O}(n \log n)$
  - Using X, Bob computes the size s of a smallest point-line cover of $\mathcal{P}$
  - He then sends s over to Alice
    - Cost: Zero
  - Alice outputs $s \overset{?}{\leq} k$
  - Total cost: $\mathcal{O}(n \log n)$

The Upper Bound

A first attempt at such a protocol

- Given an instance $(\mathcal{P}, k)$; $|\mathcal{P}| = n$ of Point-Line Cover
  - Alice computes an encoding X of $\mathcal{P}$, where X has $\mathcal{O}(n \log n)$ bits
  - She then sends X over to Bob
    - Cost: $\mathcal{O}(n \log n)$
  - Using X, Bob computes the size s of a smallest point-line cover of $\mathcal{P}$
  - He then sends s over to Alice
    - Cost: Zero
  - Alice outputs $s \overset{?}{\leq} k$
  - Total cost: $\mathcal{O}(n \log n)$
- What's missing here?
  - No known Alice-time encoding of $\mathcal{P}$ into $\mathcal{O}(n \log n)$ bits

The Upper Bound

A first attempt at such a protocol

- Given an instance $(\mathcal{P}, k)$; $|\mathcal{P}| = n$ of Point-Line Cover
  - Alice computes an encoding X of $\mathcal{P}$, where X has $\mathcal{O}(n \log n)$ bits
  - She then sends X over to Bob
    - Cost: $\mathcal{O}(n \log n)$
  - Using X, Bob computes the size s of a smallest point-line cover of $\mathcal{P}$
  - He then sends s over to Alice
    - Cost: Zero
  - Alice outputs $s \overset{?}{\leq} k$
  - Total cost: $\mathcal{O}(n \log n)$
- Our way out:
  - An Alice-time encoding of $\mathcal{P}$ which *effectively* has $\mathcal{O}(n \log n)$ bits
  - This encoding **actually** has many more bits, namely n^3

The Upper Bound

A first attempt at such a protocol

- Given an instance $(\mathcal{P}, k)$; $|\mathcal{P}| = n$ of Point-Line Cover
  - Alice computes an encoding X of $\mathcal{P}$, where X has $\mathcal{O}(n \log n)$ bits
  - She then sends X over to Bob
    - Cost: $\mathcal{O}(n \log n)$
  - Using X, Bob computes the size s of a smallest point-line cover of $\mathcal{P}$
  - He then sends s over to Alice
    - Cost: Zero
  - Alice outputs $s \stackrel{?}{\leq} k$
  - Total cost: $\mathcal{O}(n \log n)$

- Our way out:
  - An Alice-time encoding of $\mathcal{P}$ which *effectively* has $\mathcal{O}(n \log n)$ bits
  - This encoding **actually** has many more bits, namely n^3
- *Any* n-point instance of Point-Line Cover encodes to one of a set of $2^{\mathcal{O}(n \log n)}$ strings, each of length n^3
- We replace a *small* encoding with a *sparse* one

The Upper Bound

The sparse encoding

Theorem (Alon, 1986)

There is an encoding of sets of points on a plane into bit strings such that:

1. The encoding can be computed in polynomial time
2. It maps every n-point set to a bit string of length n^3
3. For each n, all these n^3-bit strings belong to a set B_n ; $|B_n| = n^{\mathcal{O}(n)}$
4. If point sets $\mathcal{P}$ and $\mathcal{Q}$ map to the same string in B_n, then they are equivalent with respect to Point-Line Cover

- The encoding is called an Abstract Order Type Representation

The Upper Bound

A protocol of cost $\mathcal{O}(n \log n)$ for Point-Line Cover

- The input instance $(\mathcal{P}, k)$; $|\mathcal{P}| = n$ is with Alice

The Upper Bound

A protocol of cost $\mathcal{O}(n \log n)$ for Point-Line Cover

- The input instance $(\mathcal{P}, k)$; $|\mathcal{P}| = n$ is with Alice
- Alice computes Alon's encoding X of $\mathcal{P}$, where $|X| = n^3$
  - She cannot send all of X over to Bob, it's too costly

The Upper Bound

A protocol of cost $\mathcal{O}(n \log n)$ for Point-Line Cover

- The input instance $(\mathcal{P}, k)$; $|\mathcal{P}| = n$ is with Alice
- Alice computes Alon's encoding X of $\mathcal{P}$, where $|X| = n^3$
  - She cannot send all of X over to Bob, it's too costly
- Alice sends the number n over to Bob
  - Cost: $\mathcal{O}(\log n)$

The Upper Bound

A protocol of cost $\mathcal{O}(n \log n)$ for Point-Line Cover

- The input instance $(\mathcal{P}, k)$; $|\mathcal{P}| = n$ is with Alice
- Alice computes Alon's encoding X of $\mathcal{P}$, where $|X| = n^3$
  - She cannot send all of X over to Bob, it's too costly
- Alice sends the number n over to Bob
  - Cost: $\mathcal{O}(\log n)$
- Using n, Bob computes a sorted list B_n of all possible encodings of n-point sets; $|B_n| = n^{\mathcal{O}(n)}$

The Upper Bound

A protocol of cost $\mathcal{O}(n \log n)$ for Point-Line Cover

- The input instance $(\mathcal{P}, k)$; $|\mathcal{P}| = n$ is with Alice
- Alice computes Alon's encoding X of $\mathcal{P}$, where $|X| = n^3$
  - She cannot send all of X over to Bob, it's too costly
- Alice sends the number n over to Bob
  - Cost: $\mathcal{O}(\log n)$
- Using n, Bob computes a sorted list B_n of all possible encodings of n-point sets; $|B_n| = n^{\mathcal{O}(n)}$
- He then sends the *median* element M of this back to Alice
  - Cost: Zero

The Upper Bound

A protocol of cost $\mathcal{O}(n \log n)$ for Point-Line Cover

- The input instance $(\mathcal{P}, k)$; $|\mathcal{P}| = n$ is with Alice
- Alice computes Alon's encoding X of $\mathcal{P}$, where $|X| = n^3$
  - She cannot send all of X over to Bob, it's too costly
- Alice sends the number n over to Bob
  - Cost: $\mathcal{O}(\log n)$
- Using n, Bob computes a sorted list B_n of all possible encodings of n-point sets; $|B_n| = n^{\mathcal{O}(n)}$
- He then sends the *median* element M of this back to Alice
  - Cost: Zero
- Alice compares M with X and tells Bob whether M is before, after, or equal to X in lexicographic order
  - Cost: A bit and a half

The Upper Bound

A protocol of cost $\mathcal{O}(n \log n)$ for Point-Line Cover

- The input instance $(\mathcal{P}, k)$; $|\mathcal{P}| = n$ is with Alice
- Alice computes Alon's encoding X of $\mathcal{P}$, where $|X| = n^3$
  - She cannot send all of X over to Bob, it's too costly
- Alice sends the number n over to Bob
  - Cost: $\mathcal{O}(\log n)$
- Using n, Bob computes a sorted list B_n of all possible encodings of n-point sets; $|B_n| = n^{\mathcal{O}(n)}$
- He then sends the *median* element M of this back to Alice
  - Cost: Zero
- Alice compares M with X and tells Bob whether M is before, after, or equal to X in lexicographic order
  - Cost: A bit and a half
- Bob throws out that half of the list B_n where X cannot be present

The Upper Bound

A protocol of cost $\mathcal{O}(n \log n)$ for Point-Line Cover

- The input instance $(\mathcal{P}, k)$; $|\mathcal{P}| = n$ is with Alice
- Alice computes Alon's encoding X of $\mathcal{P}$, where $|X| = n^3$
  - She cannot send all of X over to Bob, it's too costly
- Alice sends the number n over to Bob
  - Cost: $\mathcal{O}(\log n)$
- Using n, Bob computes a sorted list B_n of all possible encodings of n-point sets; $|B_n| = n^{\mathcal{O}(n)}$
- He then sends the *median* element M of this back to Alice
  - Cost: Zero
- Alice compares M with X and tells Bob whether M is before, after, or equal to X in lexicographic order
  - Cost: A bit and a half
- Bob throws out that half of the list B_n where X cannot be present
- He then computes the median of the remaining list, and they repeat the above procedure

The Upper Bound

A protocol of cost $\mathcal{O}(n \log n)$ for Point-Line Cover

- The input instance $(\mathcal{P}, k)$; $|\mathcal{P}| = n$ is with Alice
- Alice computes Alon's encoding X of $\mathcal{P}$, where $|X| = n^3$
  - She cannot send all of X over to Bob, it's too costly
- Alice sends the number n over to Bob
  - Cost: $\mathcal{O}(\log n)$
- Using n, Bob computes a sorted list B_n of all possible encodings of n-point sets; $|B_n| = n^{\mathcal{O}(n)}$
- He then sends the *median* element M of this back to Alice
  - Cost: Zero
- Alice compares M with X and tells Bob whether M is before, after, or equal to X in lexicographic order
  - Cost: A bit and a half
- Bob throws out that half of the list B_n where X cannot be present
- He then computes the median of the remaining list, and they repeat the above procedure
- After going back and forth for $\mathcal{O}(\log(|B_n|)) = \mathcal{O}(n \log n)$ rounds, Bob *knows* what X is

The Upper Bound

A protocol of cost $\mathcal{O}(n \log n)$ for Point-Line Cover

- The input instance $(\mathcal{P}, k)$; $|\mathcal{P}| = n$ is with Alice
- Alice computes Alon's encoding X of $\mathcal{P}$, where $|X| = n^3$
  - She cannot send all of X over to Bob, it's too costly
- Alice sends the number n over to Bob
  - Cost: $\mathcal{O}(\log n)$
- Using n, Bob computes a sorted list B_n of all possible encodings of n-point sets; $|B_n| = n^{\mathcal{O}(n)}$
- He then sends the *median* element M of this back to Alice
  - Cost: Zero
- Alice compares M with X and tells Bob whether M is before, after, or equal to X in lexicographic order
  - Cost: A bit and a half
- Bob throws out that half of the list B_n where X cannot be present
- He then computes the median of the remaining list, and they repeat the above procedure
- After going back and forth for $\mathcal{O}(\log(|B_n|)) = \mathcal{O}(n \log n)$ rounds, Bob *knows* what X is
- Using X, Bob computes the size s of a smallest point-line cover of $\mathcal{P}$

The Upper Bound

A protocol of cost $\mathcal{O}(n \log n)$ for Point-Line Cover

- The input instance $(\mathcal{P}, k)$; $|\mathcal{P}| = n$ is with Alice
- Alice computes Alon's encoding X of $\mathcal{P}$, where $|X| = n^3$
  - She cannot send all of X over to Bob, it's too costly
- Alice sends the number n over to Bob
  - Cost: $\mathcal{O}(\log n)$
- Using n, Bob computes a sorted list B_n of all possible encodings of n-point sets; $|B_n| = n^{\mathcal{O}(n)}$
- He then sends the *median* element M of this back to Alice
  - Cost: Zero
- Alice compares M with X and tells Bob whether M is before, after, or equal to X in lexicographic order
  - Cost: A bit and a half
- Bob throws out that half of the list B_n where X cannot be present
- He then computes the median of the remaining list, and they repeat the above procedure
- After going back and forth for $\mathcal{O}(\log(|B_n|)) = \mathcal{O}(n \log n)$ rounds, Bob *knows* what X is
- Using X, Bob computes the size s of a smallest point-line cover of $\mathcal{P}$
- He then sends s back to Alice
  - Cost: Zero

The Upper Bound

A protocol of cost $\mathcal{O}(n \log n)$ for Point-Line Cover

- The input instance $(\mathcal{P}, k)$; $|\mathcal{P}| = n$ is with Alice
- Alice computes Alon's encoding X of $\mathcal{P}$, where $|X| = n^3$
  - She cannot send all of X over to Bob, it's too costly
- Alice sends the number n over to Bob
  - Cost: $\mathcal{O}(\log n)$
- Using n, Bob computes a sorted list B_n of all possible encodings of n-point sets; $|B_n| = n^{\mathcal{O}(n)}$
- He then sends the *median* element M of this back to Alice
  - Cost: Zero
- Alice compares M with X and tells Bob whether M is before, after, or equal to X in lexicographic order
  - Cost: A bit and a half
- Bob throws out that half of the list B_n where X cannot be present
- He then computes the median of the remaining list, and they repeat the above procedure
- After going back and forth for $\mathcal{O}(\log(|B_n|)) = \mathcal{O}(n \log n)$ rounds, Bob *knows* what X is
- Using X, Bob computes the size s of a smallest point-line cover of $\mathcal{P}$
- He then sends s back to Alice
  - Cost: Zero
- Alice outputs $s \overset{?}{\leq} k$

The Upper Bound

A protocol of cost $\mathcal{O}(n \log n)$ for Point-Line Cover

- The input instance $(\mathcal{P}, k)$; $|\mathcal{P}| = n$ is with Alice
- Alice computes Alon's encoding X of $\mathcal{P}$, where $|X| = n^3$
  - She cannot send all of X over to Bob, it's too costly
- Alice sends the number n over to Bob
  - Cost: $\mathcal{O}(\log n)$
- Using n, Bob computes a sorted list B_n of all possible encodings of n-point sets; $|B_n| = n^{\mathcal{O}(n)}$
- He then sends the *median* element M of this back to Alice
  - Cost: Zero
- Alice compares M with X and tells Bob whether M is before, after, or equal to X in lexicographic order
  - Cost: A bit and a half
- Bob throws out that half of the list B_n where X cannot be present
- He then computes the median of the remaining list, and they repeat the above procedure
- After going back and forth for $\mathcal{O}(\log(|B_n|)) = \mathcal{O}(n \log n)$ rounds, Bob *knows* what X is
- Using X, Bob computes the size s of a smallest point-line cover of $\mathcal{P}$
- He then sends s back to Alice
  - Cost: Zero
- Alice outputs $s \overset{?}{\leq} k$
- Total cost: $\mathcal{O}(n \log n)$

The Upper Bound

A protocol of cost $\mathcal{O}(n \log n)$ for Point-Line Cover

- The input instance $(\mathcal{P}, k)$; $|\mathcal{P}| = n$ is with Alice
- Alice computes Alon's encoding X of $\mathcal{P}$, where $|X| = n^3$
  - She cannot send all of X over to Bob, it's too costly
- Alice sends the number n over to Bob
  - Cost: $\mathcal{O}(\log n)$
- Using n, Bob computes a sorted list B_n of all possible encodings of n-point sets; $|B_n| = n^{\mathcal{O}(n)}$
- He then sends the *median* element M of this back to Alice
  - Cost: Zero
- Alice compares M with X and tells Bob whether M is before, after, or equal to X in lexicographic order
  - Cost: A bit and a half
- Bob throws out that half of the list B_n where X cannot be present
- He then computes the median of the remaining list, and they repeat the above procedure
- After going back and forth for $\mathcal{O}(\log(|B_n|)) = \mathcal{O}(n \log n)$ rounds, Bob *knows* what X is
- Using X, Bob computes the size s of a smallest point-line cover of $\mathcal{P}$
- He then sends s back to Alice
  - Cost: Zero
- Alice outputs $s \overset{?}{\leq} k$
- Total cost: $\mathcal{O}(n \log n)$
- This is our second main ingredient: a protocol of cost $\mathcal{O}(n \log n)$

Open problems

- Close other such “structural” gaps in kernel bounds
- A first candidate: FEEDBACK VERTEX SET
  - We have: $\mathcal{O}(k^2)$ upper bound on #vertices and #edges
  - $\Omega(k^2)$ lower bound on #edges
  - But only: $\Omega(k)$ lower bound on #vertices
  - TODO: bridge this gap in the #vertices

Thank You!