

Kernelization

Making molehills out of mountains

A brief introduction

Geevarghese Philip

Institute Seminar Week 2010

IMSc, Chennai

17/03/2010

Outline

Introduction

A Simple Example

Polynomial Kernels

Vertex Cover

Dominating Set

Conclusion

Fixed Parameter Tractability

Introduction

- Most interesting computational problems take too long to solve.
- A “problem” asks for an input-output mapping.
 - Given a certain input, compute a certain output.
 - Interesting (and hard!) even when the output is just YES/NO.
 - NP-hardness: evidence that a problem cannot be solved in time that is polynomial in the size of the input.
 - Algorithms with super-polynomial running times are useless in practice (beyond moderate input sizes).

Fixed Parameter Tractability

Introduction

- Most interesting computational problems take too long to solve.
- But we need to solve these problems in practice, and solve them fast.
 - Finding a good answer in good time, instead of using a rough guess, is often the difference:
 - between business success and bankruptcy,
 - ★ Using the fewest container ships.
 - and even between life and death!
 - ★ Developing drugs, medical diagnostics.

Fixed Parameter Tractability

Introduction

- Most interesting computational problems take too long to solve.
- But we need to solve these problems in practice, and fast.
- Fixed-Parameter Tractability is one algorithmic approach towards such problems.
 - Many hard problems have a (natural) *parameter*
 - Some measure associated with the input.
 - Takes small values for inputs of interest.
 - E.g.: The nesting depth of programs written by humans.
 - Can we solve the problem fast *when the parameter is small*?
 - We can afford to take time
 - exponential in the value of the parameter, and
 - polynomial in the size of the input.



Fixed Parameter Tractability

Introduction

- Most interesting computational problems take too long to solve.
- But we need to solve these problems in practice, and fast.
- Fixed-Parameter Tractability is one algorithmic approach towards such problems.
- Can we solve the problem fast when the parameter is small?
 - Find algorithms that solve the problem exactly in at most $f(k) \cdot n^c$ time where
 - k is the parameter, n is the size of the input
 - $f()$ is some function of k alone, and
 - c is a constant independent of k and n .

Fixed Parameter Tractability

Introduction

- Most interesting computational problems take too long to solve.
- But we need to solve these problems in practice, and fast.
- Fixed-Parameter Tractability is one algorithmic approach towards such problems.
- Can we solve the problem fast when the parameter is small?
- A Fixed Parameter Tractable (FPT) algorithm solves the problem in at most $f(k) \cdot n^c$ time.
 - The algorithm runs in reasonable (tractable) time when the parameter k is fixed (small).

Fixed Parameter Tractability

Kernelization

- Given: A parameterized problem, the parameter being k .
- Goal: Find an FPT algorithm, one that solves the problem in at most $f(k) \cdot n^c$ time.
- Designing FPT algorithms has a different flavour compared to designing traditional algorithms
 - We need to make use of the leeway given by $f()$.
- Different techniques have been developed to help design FPT algorithms:
 - Branching, **Iterated Compression**, Color Coding, Treewidth-based methods, **Kernelization**, ...
- In this talk, we take a look at kernelization.



Fixed Parameter Tractability

Kernelization

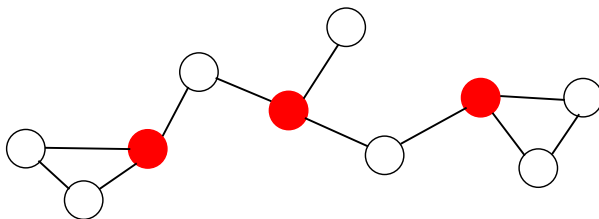
- Kernelization is (roughly speaking) the following 2-step strategy:
 1. Given an input instance I of size n , spend n^c time to “shrink” I to an **equivalent** instance I' of **small** size:
 - To solve the problem for I , it is enough to solve it for I' , and
 - The size of I' is at most $g(k)$ for some function $g()$
 2. Solve the problem for I'
 - Since $|I'| \leq g(k)$, this can be done in $h(k)$ time for some $h()$ (even if we use brute force).
- Total running time: $n^c + h(k) \leq f(k) \cdot n^c$.
- Kernelization proper is just the first, “compression” step.



Example

Planar Independent Set

- Input: A planar graph $G = (V, E)$, a positive integer k .
- Qn: Does G contain an independent set of size **at least** k ?
- Parameter: k



Example

Planar Independent Set

- Input: A planar graph $G = (V, E)$, a positive integer k .
- Qn: Does G contain an independent set of size **at least** k ?
- Parameter: k

Kernelization

- Goal: In $|G|^c$ time, convert (G, k) to (H, k') such that:
 - G has an independent set of size at least k if and only if H has one of size at least k' , and
 - $|H| \leq g(k)$ for some function $g()$

Example

Planar Independent Set

- Input: A planar graph $G = (V, E)$, a positive integer k .
- Qn: Does G contain an independent set of size **at least** k ?
- Parameter: k

Kernelization

- Goal: In $|G|^c$ time, convert (G, k) to (H, k') such that:
 - G has an independent set of size at least k if and only if H has one of size at least k' , and
 - $|H| \leq g(k)$ for some function $g()$
- How?

Example

Planar Independent Set

- Input: A planar graph $G = (V, E)$, a positive integer k .
- Qn: Does G contain an independent set of size **at least** k ?
- Parameter: k

Kernelization

- Goal: In $|G|^c$ time, convert (G, k) to (H, k') such that:
 - G has an independent set of size at least k if and only if H has one of size at least k' , and
 - $|H| \leq g(k)$ for some function $g()$
- How?
- Use the Four Colour Theorem!

Example

Planar Independent Set

- Input: A planar graph $G = (V, E)$, a positive integer k .
- Qn: Does G contain an independent set of size **at least** k ?
- Parameter: k

Kernelization

- The Four Colour Theorem (Appel and Haken, 1976): Four colours are sufficient to colour all the vertices of a planar graph in such a way that no two adjacent vertices have the same colour.

Example

Planar Independent Set

- Input: A planar graph $G = (V, E)$, a positive integer k .
- Qn: Does G contain an independent set of size **at least** k ?
- Parameter: k

Kernelization

- The Four Colour Theorem (Appel and Haken, 1976): Four colours are sufficient to colour all the vertices of a planar graph in such a way that no two adjacent vertices have the same colour.
- Corollary: Any planar graph G on n vertices contains an independent set of size at least $n/4$.
 - Look at any four-colouring of G . At least one colour class contains $n/4$ or more vertices.
 - Each colour class is an independent set, by definition.

Example

Planar Independent Set

- Input: A planar graph $G = (V, E)$, a positive integer k .
- Qn: Does G contain an independent set of size **at least** k ?
- Parameter: k

Kernelization

- G contains an independent set of size at least $|V|/4$.
 - If $k \leq |V|/4$, then the answer is YES.
 - Otherwise, $|V| < 4k$, and so the size of the input graph G is bounded by a linear function of k .
 - Set $H = G, k' = k$
- All this takes $O(|G|)$ time (just the time to find $|V|$).

Recapitulating...

- An instance of a parameterized problem is a pair (I, k) ; k is the *parameter*.
- An FPT algorithm solves the problem in $f(k) \cdot |I|^c$ time.
- A kernelization algorithm for the problem
 - Takes (I, k) as input, runs in *poly* $(|I|)$ time, and outputs an equivalent instance (I', k') such that
 - $|I'| \leq g(k)$, and
 - $k' \leq h(k)$
 - (I', k') is called the *kernel*.
- Planar Independent Set has a kernel on at most $4k$ vertices.

Kernelization

Polynomial Kernels

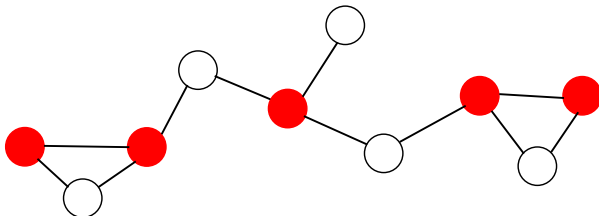
- Theorem (folklore): A parameterized problem has an FPT algorithm if and only if it has a kernelization algorithm.
 - The proof implies, given an $f(k) \cdot \text{poly}(|I|)$ FPT algorithm, a kernel of size $f(k)$.
- Given a parameterized problem, we are interested in finding kernels of size $\text{poly}(k)$.
 - We want to shrink the input instance in *polynomial* time to a size that is *polynomial* in the parameter.
 - We achieved this for Planar Independent Set: $< 4k$ vertices, $< 12k - 6$ edges.



Example

Vertex Cover

- Input: A graph $G = (V, E)$, a positive integer k .
- Qn: Does G have a vertex cover of size **at most** k ?
- Parameter: k
- A set of **vertices** that cover (touch) all **edges**.



Example

Vertex Cover

- Input: A graph $G = (V, E)$, a positive integer k .
- Qn: Does G have a vertex cover of size **at most** k ?
- Parameter: k

- A simple FPT algorithm that runs in $O(2^k \cdot n^2)$ time.
 - Yields a kernel of size $O(2^k)$.
- We can do better!

Vertex Cover

A Quadratic Kernel

Buss' kernelization (1993)

- Throw out all vertices with degree zero:
 - These do not cover any edge.
- If a vertex v has degree more than k , then pick v in the solution:
 - Either v or *all* its neighbours must be in in *any* vertex cover.
- In the remaining graph, each vertex has degree at most k .
- If there are more than k^2 edges left, then say NO
 - k vertices of degree at most k can cover at most k^2 edges
- Otherwise, the graph has at most k^2 edges (and $k^2 + k$ vertices): an $O(k^2)$ -size kernel.

Vertex Cover

A “Linear” Kernel

- It gets even better!

Vertex Cover

A “Linear” Kernel

- It gets even better!
 - There is a kernel on at most $2k$ vertices and $O(k^2)$ edges (Chen et. al, 2001),
 - And I’m not going to talk about it.

Vertex Cover

A “Linear” Kernel

- It gets even better!
 - There is a kernel on at most $2k$ vertices and $O(k^2)$ edges (Chen et. al, 2001),
 - And I’m not going to talk about it.
- This implies something quite remarkable:

Vertex Cover

A “Linear” Kernel

- It gets even better!
 - There is a kernel on at most $2k$ vertices and $O(k^2)$ edges (Chen et. al, 2001),
 - And I’m not going to talk about it.
- This implies something quite remarkable:
- If you give me a graph G on a 1000 vertices, and ask me if G has a vertex cover of size at most 20, in time *linear* in the size of G , I can
 - either tell you NO, or
 - come up with a graph H on **at most 40 vertices**; solving the problem on H solves the problem on G .



Vertex Cover

A “Linear” Kernel

- It gets even better!
 - There is a kernel on at most $2k$ vertices and $O(k^2)$ edges (Chen et. al, 2001),
 - And I’m not going to talk about it.
- This implies something quite remarkable:
- If you give me a graph G on a 10000 vertices, and ask me if G has a vertex cover of size at most 20, in time *linear* in the size of G , I can
 - either tell you NO, or
 - come up with a graph H on **at most 40 vertices**; solving the problem on H solves the problem on G .

Vertex Cover

A “Linear” Kernel

- It gets even better!
 - There is a kernel on at most $2k$ vertices and $O(k^2)$ edges (Chen et. al, 2001),
 - And I’m not going to talk about it.
- This implies something quite remarkable:
- If you give me a graph G on a 100000 vertices, and ask me if G has a vertex cover of size at most 20, in time *linear* in the size of G , I can
 - either tell you NO, or
 - come up with a graph H on **at most 40 vertices**; solving the problem on H solves the problem on G .

Vertex Cover

A “Linear” Kernel

- It gets even better!
 - There is a kernel on at most $2k$ vertices and $O(k^2)$ edges (Chen et. al, 2001),
 - And I’m not going to talk about it.
- This implies something quite remarkable:
- If you give me a graph G on a 1000000 vertices, and ask me if G has a vertex cover of size at most 20, in time *linear* in the size of G , I can
 - either tell you NO, or
 - come up with a graph H on **at most 40 vertices**; solving the problem on H solves the problem on G .



Vertex Cover

A “Linear” Kernel

- It gets even better!
 - There is a kernel on at most $2k$ vertices and $O(k^2)$ edges (Chen et. al, 2001),
 - And I’m not going to talk about it.
- This implies something quite remarkable:
- If you give me a graph G on a 10000000 vertices, and ... well, you get the drift.

Dominating Set

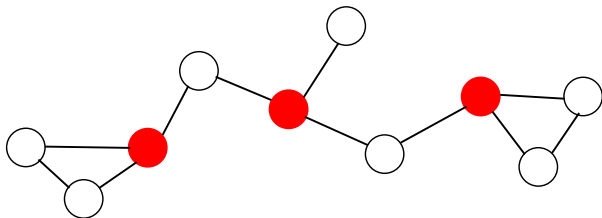
On restricted graph classes

- Recall the Planar Independent Set problem for which we got a “cheat” kernel on at most $4k$ vertices
 - Using the Four Colour Theorem as a crutch.
- The same problem, on *general* graphs, is not even Fixed Parameter Tractable*
 - So no hope of obtaining a kernel of *any* size, much less a polynomial-size kernel.

Dominating Set

On restricted graph classes

- The Dominating Set problem is also not Fixed Parameter Tractable on general graphs*
 - Input: A graph $G = (V, E)$, a positive integer k .
 - Qn: Does G contain a dominating set of size **at most** k ?
 - Parameter: k



Dominating Set

On restricted graph classes

- The problem does not have any kernel on general graphs*.
- Does the problem have a polynomial kernel on planar graphs?
- No easy way out! Open for long.
- In a major breakthrough, a linear kernel (on $335k$ vertices!) was obtained Alber et. al. in 2004.
- Improved later to $67k$ by Chen et. al.

Dominating Set

On restricted graph classes

- Does the problem have polynomial kernels on larger classes of graphs?

Dominating Set

On restricted graph classes

- Does the problem have polynomial kernels on larger classes of graphs?
- Results for larger classes:
 - Genus- g graphs – $O(k + g)$ -vertex kernel – Fomin et. al., 2004.
 - H -minor-free graphs – $poly(k)$ -vertex kernel – Alon et.al., 2008.

Dominating Set

On restricted graph classes

- Does the problem have polynomial kernels on larger classes of graphs?
- Results for larger classes:
 - Genus- g graphs – $O(k + g)$ -vertex kernel – Fomin et. al., 2004.
 - H -minor-free graphs – $poly(k)$ -vertex kernel – Alon et.al., 2008.
- Our result (Philip, Raman, Sikdar, 2009.):
 - The Dominating Set problem has polynomial kernels on
 - graphs of bounded degeneracy, and
 - graphs that do not have a fixed complete bipartite graph as a subgraph.
- Subsumes and is strictly larger than all the other graph classes for which polynomial kernels are known for this problem.
 - E.g., planar graphs do not have $K_{3,3}$ as a subgraph.

Summary

- A kernelization algorithm is a polynomial-time algorithm that takes an input instance (I, k) of a parameterized problem and outputs an *equivalent* instance (the *kernel*) whose size and parameter are bounded by functions of k .
- A parameterized problem has a kernel if and only if it has an FPT algorithm.
- Finding polynomial-sized kernels is both theoretically and practically important.
- Polynomial kernels have been found for many parameterized problems: we saw three examples.
- Recent theoretical advances (Bodlaender et. al., 2008) provide tools to demonstrate the non-existence* of polynomial kernels.

Thank You!