FEDOR V. FOMIN

Kernelization Algorithms

LNMB conference, Luntheren

15.01.2013

Preprocessing

- Preprocessing: reducing the input to something simpler...

Preprocessing

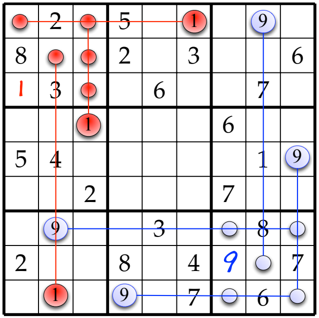
- Preprocessing: reducing the input to something simpler...
- Kernelization: an attempt to analyze preprocessing...

Preprocessing. Sudoku

	2		5		1		9	
8			2		3			6
	3			6			7	
		1				6		
5	4						1	9
		2				7		
	9			3			8	
2			8		4			7
	1		9		7		6	

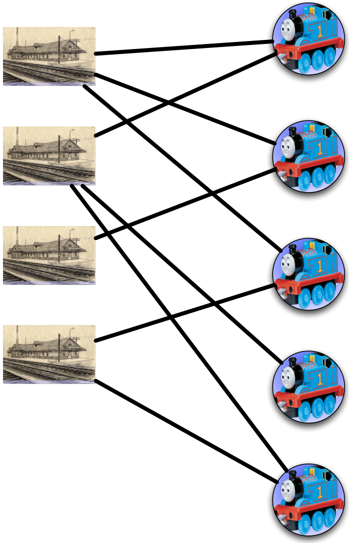
4	2	*6*	5	*7*	1	*3*	9	*8*
8	*5*	*7*	2	*9*	3	*1*	*4*	6
1	3	*9*	*4*	6	*8*	*2*	7	*5*
9	*7*	1	*3*	*8*	*5*	6	*2*	*4*
5	4	*3*	*7*	*2*	*6*	*8*	1	9
6	*8*	2	*1*	*4*	*9*	7	*5*	*3*
7	9	*4*	*6*	3	*2*	*5*	8	*1*
2	*6*	*5*	8	*1*	4	*9*	*3*	7
3	1	*8*	9	*5*	7	*4*	6	*2*

Preprocessing. Sudoku



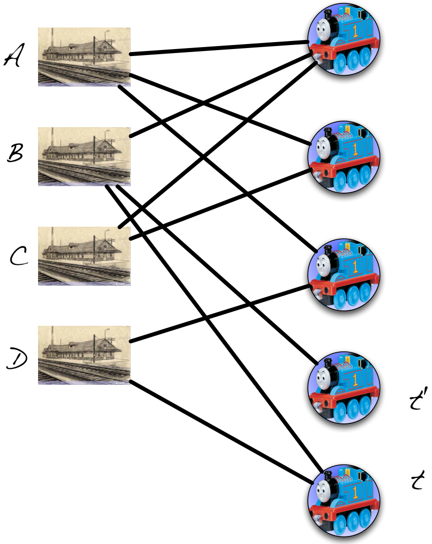
Cross-hatching rule

Preprocessing. Train management



- We want a minimum-sized subset of stations S such that every train stops at least at one station from S

Preprocessing. Train management



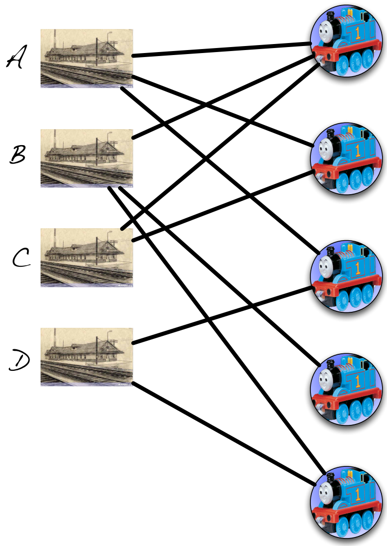
Reduction Rules:

- Rule 1 If $S(t) \subseteq S(t')$, then remove t

$S(t)=\{B,D\}$

$S(t')=\{B\}$

Preprocessing. Train management



Reduction Rules:

- Rule 2 If $T(s) \subseteq T(s')$, then remove s

Preprocessing. Train management

K.Weihe, ALEX, 1998: Similar preprocessing for real-world data from the German and European train schedules (25.000 stations, 154.000 trains and 160.000 single train stops) the data reduction merely took a few minutes to reduce the original, huge input graph into a graph consisting of disjoint components of size at most 50.

Preprocessing is ubiquitous

- Commercial linear program solvers like CPLEX
- Navigation systems
- Microarray data analysis for the classification of cancer types
- ...

Analysis of Algorithms

- Powerful tools developed since 1960s
- Theory of NP-completeness

Analysis of Algorithms. Naive question

- Take your favourite NP-complete problem
- Is there preprocessing algorithm that guarantees to reduce every instance of your problem, say by 5%?

Analysis of Algorithms. Naive question

- Take your favourite NP-complete problem
- Is there preprocessing algorithm that guarantees to reduce every instance of your problem, say by 5%?
- That would be very strange!!!

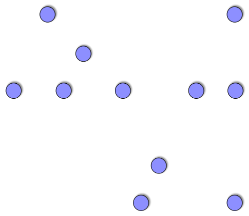
Theory of Computing for the 21st century

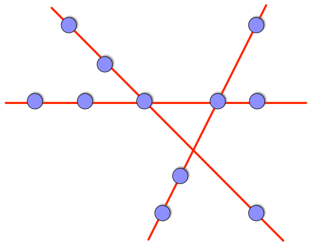
"While theoretical work on models of computation and methods for analyzing algorithms has had enormous payoff, we are not done. In many situations, simple algorithms do well. We don't understand why! It is apparent that worst-case analysis does not provide useful insights on the performance of algorithms and heuristics and our models of computation need to be further developed and refined."

Condon, Edelsbrunner, Emerson, Fortnow, Haber, Karp, Leivant, Lipton, Lynch, Parberry, Papadimitriou, Rabin, Rosenberg, Royer, Savage, Selman, Smith, Tardos, and Vitter, *Challenges for theory of computing: Report for an NSF-sponsored workshop on research in theoretical computer science*, 1999.

Anther try: Covering Points with Lines

Task: Given a set P of n points in the plane and an integer k, find k lines that cover all the points.





Note: We can assume that every line of the solution covers at least 2 points, thus there are at most n^2 candidate lines.

Covering Points with Lines

Reduction Rule If there is a line L covering more than k points, remove all points covered by L and reduce the parameter k by one.

Covering Points with Lines

Reduction Rule If there is a line L covering more than k points, remove all points covered by L and reduce the parameter k by one.

Why this rule is sound?

Covering Points with Lines

At every step of Reduction Rule we obtain a problem with a smaller number of points. Thus we

- either end up with the problem with no points left, and in this case we solved the problem; YES!

Covering Points with Lines

At every step of Reduction Rule we obtain a problem with a smaller number of points. Thus we

- either end up with the problem with no points left, and in this case we solved the problem; YES!
- or the parameter k is zero but some points are left, in this case the problem does not have solution; NO!

Covering Points with Lines

At every step of Reduction Rule we obtain a problem with a smaller number of points. Thus we

- either end up with the problem with no points left, and in this case we solved the problem; YES!
- or the parameter k is zero but some points are left, in this case the problem does not have solution; NO!
- or we arrive at the problem for which our Reduction Rule cannot be applied. What happens here?

Covering Points with Lines

Reduction Rule If there is a line L covering more than k points, remove all points covered by L and reduce the parameter k by one.

If Rule cannot be applied, WE HAVE AT MOST k^2 POINTS!

We have an algorithm that

Input: An instance of our (NP-complete) problem of size n, and a parameter k.
Output: Either correct solution, or an equivalent instance.
Properties of the algorithm and the reduced instance

- It runs in time $O(n^2)$ [polynomial time]
- Outputs an equivalent instance of size k^2 [the size of reduced instance depends only on parameter k].

What happens?

Preprocessing: We are not able to say how much the whole input length has been changed but we can analyse progress of preprocessing in terms of some parameter.

Parameterized complexity

Main idea: Instead of expressing the running time as a function $T(n)$ of n, we express it as a function $T(n, k)$ of the input size n and some parameter k of the input.

In other words: we do not want to be efficient on all inputs of size n, only for those where k is small.

Parameterized complexity

What can be the parameter k?

- The size k of the solution we are looking for.
- The maximum degree of the input graph.
- The diameter of the input graph.
- The length of clauses in the input Boolean formula.
- $\dots$

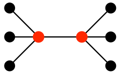
Fixed-parameter tractability

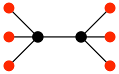
Main definition:

A parameterized problem is **fixed-parameter tractable (FPT)** if there is an $f(k)n^c$ time algorithm for some constant c solving this problem.

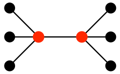
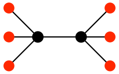
Parameterized complexity

Problem:	MIN VERTEX COVER	MAX INDEPENDENT SET
Input:	Graph G, integer k	Graph G, integer k
Question:	Is it possible to cover the edges with k vertices?	Is it possible to find k independent vertices?

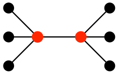
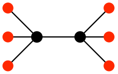




Parameterized complexity

Problem:	MIN VERTEX COVER	MAX INDEPENDENT SET
Input:	Graph G, integer k	Graph G, integer k
Question:	Is it possible to cover the edges with k vertices?	Is it possible to find k independent vertices?
		
Complexity:	NP-complete	NP-complete

Parameterized complexity

Problem:	MIN VERTEX COVER	MAX INDEPENDENT SET
Input:	Graph G, integer k	Graph G, integer k
Question:	Is it possible to cover the edges with k vertices?	Is it possible to find k independent vertices?
		
Complexity:	NP-complete	NP-complete
Complete Enumeration:	$O(n^k)$ possibilities	$O(n^k)$ possibilities
	$O(2^k n^2)$ algorithm exists	No $n^{o(k)}$ algorithm known

FPT problems

Examples of NP-hard problems that are FPT:

- Finding a vertex cover of size k.
- Finding a path of length k.
- Finding k disjoint triangles.
- Drawing the graph in the plane with k edge crossings.
- Finding disjoint paths that connect k pairs of points.
- $\ldots$

FPT problems

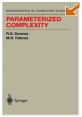
Examples of NP-hard problems that are FPT:

- Finding a vertex cover of size k.
- Finding a path of length k.
- Finding k disjoint triangles.
- Drawing the graph in the plane with k edge crossings.
- Finding disjoint paths that connect k pairs of points.
- ...

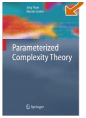
Companion Complexity Theory.
Examples of W-hard problems:

- Finding a dominating set of size k.
- Finding a clique of size k.
- Finding an independent set of size k.
- ...

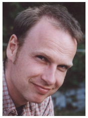
Books

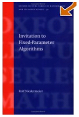

MONOGRAPHS IN COMPUTER SCIEN
PARAMETERIZED COMPLEXITY
R.G. Downey
M.R. Fellows
Springer

Downey-Fellows: Parameterized Complexity, Springer, 1999


Jörg Flum
Martin Grohe
Parameterized Complexity Theory
Springer

Flum-Grohe: Parameterized Complexity Theory, Springer, 2006




Invitation to Fixed-Parameter Algorithms
Rolf Niedermeier

Niedermeier: Invitation to Fixed-Parameter Algorithms, Oxford University Press, 2006.

Coming back to preprocessing

Kernelization algorithm:

Input: An instance of parameterized problem with parameter k.
Output: An equivalent instance of size at most $f(k)$, where $f(k)$ is some function of parameter k only.

Properties of the kernelization algorithm

- Algorithm runs in polynomial time.

$f(k)$: size of the kernel

Covering points by lines

Our algorithm for COVERING POINTS WITH LINES is a kernelization algorithm:

On input with n vertices and parameter k, it produces in polynomial time an equivalent instance of size at most k^2.

Example: Vertex Cover

- ILP for Vertex Cover

$$\begin{aligned} \text{Minimize} \quad & \textstyle\sum_{v \in V} x_v \\ \text{subject to} \quad & x_u + x_v \geq 1, \quad \forall e = \{u, v\} \in E \\ & x_v \in \{0, 1\}, \quad \forall v \in V \end{aligned}$$

- LP Relaxation

$$\begin{aligned} \text{Minimize} \quad & \textstyle\sum_{v \in V} x_v \\ \text{subject to} \quad & x_u + x_v \geq 1, \quad \forall e = \{u, v\} \in E \\ & 0 \leq x_v \leq 1, \quad \forall v \in V \end{aligned}$$

Example: Vertex Cover

Theorem (Nemhauser-Trotter, 1975)

LP relaxation of IPL vertex cover has a superoptimal half-integral $\{0, 1/2, 1\}$ *solution.*

Corollary

Parameterized Vertex Cover admits a kernel with at most $2k$ *vertices and* $O(k^2)$ *edges, i.e. a kernel of size* $O(k^2)$*.*

Example: Vertex Cover

Theorem (Nemhauser-Trotter, 1975)

LP relaxation of IPL vertex cover has a superoptimal half-integral $\{0, 1/2, 1\}$ *solution.*

Corollary

Parameterized Vertex Cover admits a kernel with at most $2k$ *vertices and* $O(k^2)$ *edges, i.e. a kernel of size* $O(k^2)$*.*

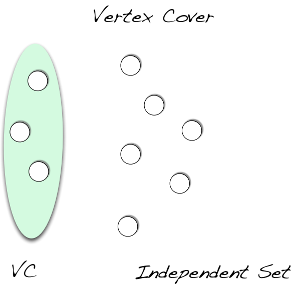
Proof.

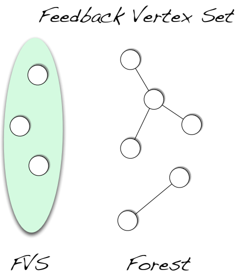
By Nemhauser-Trotter, LP relaxation gives a graph on at most $2k$ vertices. $\square$

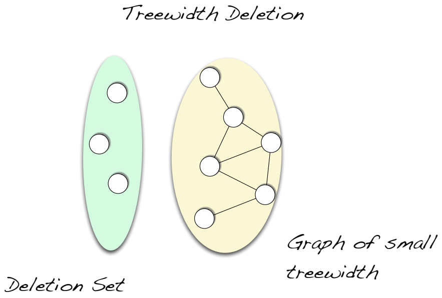
Example: Feedback Vertex Set

Theorem (Thomasse, SODA 2009)

Parameterized Feedback Vertex Set problem admits a kernel with $O(k^2)$ vertices and $O(k^2)$ edges, i.e. of size $O(k^2)$.


Vertex Cover
VC
Independent Set


Feedback Vertex Set
FVS
Forest


Treewidth Deletion
Deletion Set
Graph of small treewidth

Example: Treewidth Deletion Set

Theorem (FF, Lokshtanov, Misra, Saurabh, FOCS 2012)

Parameterized Treewidth Deletion problem admits a polynomial kernel.

Origin of kernelization

Parameterized Complexity

Folklore: A parameterized problem admits kernelization if and only if it is fixed parameter tractable.

Origin of kernelization

Thus the powerful tools of Parameterized Complexity can be used to decide if a problem has a kernel or not.

Origin of kernelization

Thus the powerful tools of Parameterized Complexity can be used to decide if a problem has a kernel or not.

However, Parameterized Complexity cannot be used to estimate size of the kernel (the function $f(k)$ bounding the output of the algorithm).

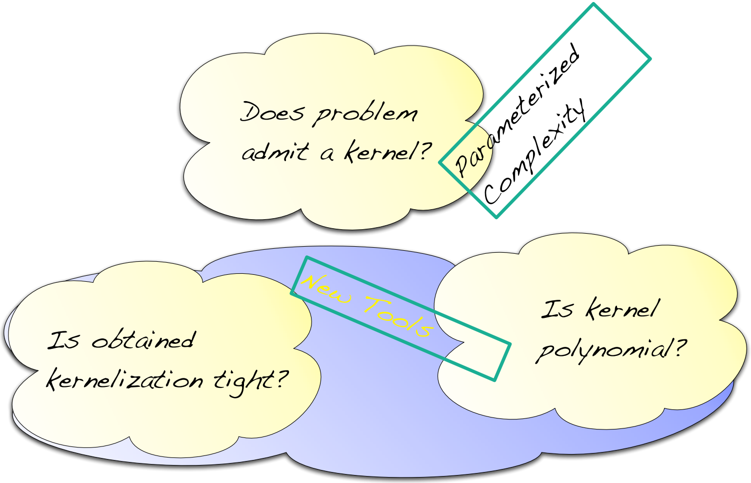
Questions about kernelization

Does problem admit a kernel?

Is obtained kernelization tight?

Is kernel polynomial?

Questions about kernelization


Does problem admit a kernel?
Parameterized Complexity
New Tools
Is obtained kernelization tight?
Is kernel polynomial?

Polynomial time algorithms

- Exact
- Approximation

Polynomial time algorithms

- Exact
- Approximation
- Kernelization

Why kernelization?

- Hope to shed some light on the effectiveness of established heuristics
- Help to design better heuristics in mathematically disciplined ways
- Deeper understanding of what can be done efficiently with intractable computational problems

Few years ago

The Lost Continent of Polynomial Time: Preprocessing and Kernelization

Michael R. Fellows

School of Electrical Engineering and Computer Science
University of Newcastle, University Drive, Callaghan NSW 2308, Australia
`mfellows@cs.newcastle.edu.au`

Abstract. One of the main objectives of the talk is to survey the history of the practical algorithmic strategy of *preprocessing* (also called *data-reduction* and *kernelization*) since the beginnings of computer science, and to overview what theoretical computer science has been able to say about it.

Thriving research area

- Complexity
- Algorithms

Complexity

- There are problems that have exponential kernels and (unless polynomial hierarchy collapses to the third level) have no polynomial kernels.
- The work of Fortnow and Santhanam (STOC 2008) and of Bodlaender, Downey, Fellows, and Hermelin (ICALP 2008) on OR-expressive problems

Theorem

The problem of finding a path of length at least k has no polynomial kernel unless polynomial hierarchy collapses to the third level.

Complexity. More OR-expressive problems

- k-Path, k-Cycle, k-Exact Cycle and k-Short Cheap Tour,
- k-Graph Minor Order Test and k-Bounded Treewidth Subgraph Test,
- k-Planar Graph Subgraph Test and k-Planar Graph Induced Subgraph Test

Complexity. AND-expressive problems

Drucker (FOCS, 2012)

- k-Cutwidth, k-Search Number,
- k-Pathwidth, k-Treewidth, and k-Branchwidth

Complexity

- Extension of technique by Lokshtanov and Saurabh (ICALP 2009): Steiner tree, connected vertex cover
- More by Bodlaender, Jansen, and Kratsch (STACS 2010)

Complexity

- Dell and van Melkebeek (STOC 2010): lower bounds on polynomial sizes of kernels.

Theorem

The problem of finding a vertex cover at most k has no kernel with $k^{(2-\varepsilon)}$ edges for any $\varepsilon > 0$ unless polynomial hierarchy collapses to the third level.

Note: as we discussed, kernel with k^2 edges can be constructed.

Algorithms

- Meta-algorithmic results on sparse graphs (Bodlaender, FF, Lokshtanov, Penninks, Saurabh, Thilikos, FOCS 2009, SODA 2010): Many problems like Dominating Set, r-Dominating Set, Vertex Cover, Connected Vertex Cover, Connected Dominating Set, Almost Outerplanar, Feedback Vertex Set, Cycle Domination, different packing and covering problems... admit linear kernels on H-minor-free graphs

Algorithms

- Graph Minors Techniques (FF, Lokshtanov, Misra, Saurabh, FOCS 2012)
- Matroids (Kratsch and Walstrom, FOCS 2012)

Algorithms

Combinatorial tools

Matchings [Chor, Fellows and Juedes, 2004]

Expansion [Thomasse, 2009]

Matroid Mathching [Lokshtanov and Saurabh, 2009]

Modular and Tree decompositions

Clustering [Guo 2009, Cao and Chen 2010]

Hitting minors [FF, Lokshtanov, Misra, Philip, and Saurabh, 2011]

Meta theorems

Planar Graphs [Bodlaender et al. 2009]

H-minor free graphs [FF, Lokshtanov, Saurabh, and Thilikos, 2010]

Algebraic and Probabilistic methods

Above guarantee [Alon et al. 2010], [Gutin et al. 2011]

Matroids

Matroid representations [Kratsch and Wahlström, 2012]

Future directions

- General techniques (what are analogues for, say, LP or PCP for kernelization?)
- Approximation and kernelization

Conclusion

- Kernelization can be thought of as a polynomial-time preprocessing before attacking the problem with whatever method we have. "It does no harm" to try kernelization.
- Some kernelizations use lots of simple reduction rules and require a complicated analysis to bound the kernel size...
- ... while other kernelizations are based on surprising nice tricks and deep mathematical ideas.
- Possibility to prove lower bounds.

Further reading

Hans L. Bodlaender, Rodney G. Downey, Michael R. Fellows, and Danny Hermelin.
On problems without polynomial kernels.
J. Comput. Syst. Sci., 75(8):423–434, 2009.

Jiong Guo and Rolf Niedermeier.
Invitation to data reduction and problem kernelization.
SIGACT News, 38(1):31–45, 2007.

Neeldhara Misra, Venkatesh Raman, and Saket Saurabh.
Lower bounds on kernelization.
Discrete Optim., 8(1):110–128, 2011.