Kernelization Algorithms

Fedor V. Fomin*

Department of Informatics, University of Bergen, N-5020 Bergen, Norway
fomin@ii.uib.no

1 Introduction

Preprocessing or data reductions means reducing the input to something simpler by solving an easy part of the input and this is the type of algorithms used in almost every application. In spite of wide practical applications of preprocessing, a systematic theoretical study of such algorithms remains elusive. The framework of parameterized complexity can be used as an approach to analyse preprocessing algorithms. Input to parameterized algorithms include a parameter (in addition to the input) which is likely to be small, and this resulted in a study of preprocessing algorithms that reduce the size of the input to a pure function of the parameter (independent of the input size). Such type of preprocessing algorithms are called kernelization algorithms.

In the talk we discuss some of the classical and recent algorithmic techniques for obtaining kernels. We do not try to give a comprehensive overview of all significant results in the area—doing this will require at least a book. We refer to the surveys of Fellows [4] and Guo and Niedermeier [4, 6] for further reading on kernelization algorithms. We also do not discuss here techniques for deriving lower bounds on the sizes of the kernels. We refer to fairly comprehensive survey of Misra et al. [7] on the kernelization intractability.

Examples of kernelization: In parameterized complexity each problem instance comes with a parameter k. As a warm-up, let us consider the following parameterized examples. Our first example is about vertex cover. A set of vertices S in a graph is a vertex cover if every edge of the graph contains at least one vertex from S. In the parameterized version of vertex cover, we call it p-VERTEX COVER, we use $p-$ to emphasise that this is the parameterized problem, the parameter is integer k and we ask if the given graph has a vertex cover of size k. Another problem, p-LONGEST PATH asks if a given graph contains a path of length at least k. And finally, p-DOMINATING SET is to decide if a given graph has a dominating set of size k, i.e. a set of vertices such that every vertex of the input graph is either in this set or is adjacent to some vertex from the set.

The parameterized problem is said to admit a *kernel* if there is an algorithm that reduces the input instance down to an instance with size bounded by some function $h(k)$ of k only, while preserving the answer. The running time of this

* Supported by the European Research Council (ERC) advanced grant PREPROCESSING, reference 267959.

algorithm should be polynomial in the input size and the degree of polynomial is independent of the parameter k. Such an algorithm is called a *kernelization* algorithm. If function $h(k)$ is polynomial in k, then we say that the problem admits a polynomial kernel.

In our examples, p-Vertex Cover admits a polynomial kernel—there is a polynomial time algorithm that for any instance (G, k) of the problem outputs a new instance (G', k') such that G' has at most $2k$ vertices and G has a vertex cover at most k if and only if G' has a vertex cover of size at most k' [2]. The second example, p-Longest Path, admits a kernel but the bounding function $h(k)$ is exponential. It is possible to show that up to some assumptions from complexity theory, the problem does not admit a polynomial kernel [1]. The problem does not admit a polynomial kernels even when the input graph G is planar. Finally, p-Dominating Set admits no kernel unless FPT=W[2], the collapse of several levels in parameterized complexity hierarchy [3]. However, on planar graph p-Dominating Set admits kernel with function $h(k) = \mathcal{O}(k)$, i.e. a linear kernel.

2 Basic Definitions

Here we mainly follow notations from the book of Flum and Grohe [5]. We describe decision problems as languages over a finite alphabet Σ.

Definition 1. *Let Σ be a finite alphabet.*

(1) A parameterization *of Σ^* is a polynomial time computable mapping $\kappa : \Sigma^* \to \mathbb{N}$.*
(2) A parameterized problem *(over Σ) is a pair (Q, κ) consisting of a set $Q \subseteq \Sigma^*$ of strings over Σ and a parameterization κ of Σ^*.*

For a parameterized problem (Q, κ) over alphabet Σ, we call the strings $x \in \Sigma^*$ the *instances* of Q or (Q, κ) and the number of $\kappa(x)$ the corresponding *parameters*. We usually represent a parameterized problem in the form

Instance:	$x \in \Sigma^*$.
Parameter:	$\kappa(x)$.
Problem:	Decide whether $x \in Q$.

Very often the parameter is also a part of the instance. For example, consider the following parameterized version of the minimum feedback vertex set problem, where the instance consists of a graph G and a positive integer k, the problem is to decide whether G has a feedback vertex set, a set of vertices which removal destroys all cycles in the graph, of k elements.

p-Feedback Vertex Set

Instance:	A graph G, and a non-negative integer k.
Parameter:	k.
Problem:	Decide whether G has a feedback vertex set with at most k elements.

In this problem the instance is the string (G,k) and $\kappa(G,k) = k$. When the parameterization κ is defined as $\kappa(x,k) = k$, the paramerized problem can be defined as subsets of $\Sigma^* \times \mathbb{N}$. Here the parameter is the second component of the instance. In this survey we use both notations for parameterized problems.

The notion of kernelization is intimately linked with the notion of fixed-parameter tractability. Fixed-parameter tractable algorithms are a class of exact algorithms where the exponential blowup in the running time is restricted to a small parameter associated with the input size. That is, the running time of such an algorithm on an input of size n is of the form $\mathcal{O}\left(f\left(k\right) n^c\right)$, where k is a parameter that is typically small compared to n, $f\left(k\right)$ is a (typically super-polynomial) function of k that does not involve n, and c is a constant. Formally,

Definition 2. *A parameterized problem (Q,κ) is* fixed-parameter tractable *if there exists an algorithm that decides in $f\left(\kappa(x)\right) \cdot n^{\mathcal{O}(1)}$ time whether $x \in Q$, where $n := |x|$ and f is a computable function that does not depend on n. The algorithm is called a* fixed parameter algorithm *for the problem. The complexity class containing all fixed parameter tractable problems is called FPT.*

There is also a hierarchy of intractable parameterized problem classes above FPT, the main ones are:

$$FPT \subseteq M[1] \subseteq W[1] \subseteq M[2] \subseteq W[2] \subseteq \cdots \subseteq W[P] \subseteq XP$$

The principal analogue of the classical intractability class NP is $W[1]$, which is a strong analogue, because a fundamental problem complete for $W[1]$ is the k-Step Halting Problem for Nondeterministic Turing Machines (with unlimited nondeterminism and alphabet size) — this completeness result provides an analogue of Cook's Theorem in classical complexity. A convenient source of $W[1]$-hardness reductions is provided by the result that p-Clique is complete for $W[1]$. Other highlights of the theory include that p-Dominating Set, by contrast, is complete for $W[2]$. Another highlight is that $FPT = M[1]$ if and only if the *Exponential Time Hypothesis* fails [5]. The classical reference on Parameterized Complexity is the book of Downey and Fellows [3]. For more updated material we refer to books of Flum and Grohe [5] and Niedermeier [8].

The notion of *kernelization* is formally defined as follows.

Definition 3. *Let (Q,κ) be a parameterized problem over a finite alphabet Σ. A* kernelization algorithm*, or in short, a* kernelization*, for (Q,κ) is an algorithm that for any given $x \in \Sigma^*$ outputs in time polynomial in $|x| + \kappa(x)$ a string $x' \in \Sigma^*$ such that*

$$(x \in Q \iff x' \in Q) \text{ and } |x'|, |\kappa(x')| \leq h(\kappa(x)),$$

where h is an arbitrary computable function. If K is a kernelization of (Q,κ), then for every instance x of Q the result of running K on input x is called the kernel of x (under K). The function h is referred to as the size *of the kernel. If h is a polynomial function then we say the kernel is polynomial.*

We often say that a problem (Q, κ) admits a kernel of size h, meaning that every instance of Q has a kernel of size h. We also often say that (Q, κ) admits a kernel with property Π, meaning that every instance of Q has a kernel with property Π. For example, by saying p-Vertex Cover admits a kernel with $\mathcal{O}(k)$ vertices and $\mathcal{O}(k^2)$ edges, we mean that there is a kernelization algorithm K, such that for every instance (G, k) of the problem, there is a kernel with $\mathcal{O}(k)$ vertices and $\mathcal{O}(k^2)$ edges.

It is easy to see that if a decidable problem admits kernelization for some function f, then the problem is FPT—for every instance of the problem we run polynomial time kernelization algorithm and then use the decision algorithm to identify if this is the valid instance. Since the size of the kernel is bounded by some function of the parameter, the running time of the decision algorithm depends only on the parameter. Interestingly, the converse also holds, that is, if a problem is FPT then it admits a kernelization. The proof of this fact is quite simple, and we present it here.

Lemma 1 (Folklore, [5, 8]). *If a parameterized problem* (Q, κ) *is* FPT *then it admits kernelization.*

Proof. Suppose that there is an algorithm deciding if $x \in Q$ in time $f(\kappa(x))|x|^c$ time for some function f and constant c. If $|x| \geq f(\kappa(x))$, then we run the decision algorithm on the instance in time $f(\kappa(x))|x|^c \leq |x|^{c+1}$. If the decision algorithm outputs YES, the kernelization algorithm outputs a constant size YES instance, and if the decision algorithm outputs NO, the kernelization algorithm outputs a constant size NO instance. On the other hand, if $|x| < f(\kappa(x))$, then the kernelization algorithm outputs x. This yields a kernel of size $f(\kappa(x))$ for the problem.

Lemma 1 shows that kernelization can be seen as an alternative definition of fixed parameter tractable problems. However, we are interested in kernels that are as small as possible, and a kernel obtained using Lemma 1 has size that equals the dependence on k in the running time of the best known FPT algorithm for the problem. The question is—can we do better? The answer is that quite often we can. In fact, for many problems we can polynomial kernels. In this talk we survey some of the old and new techniques for showing that problems admit polynomial kernels.

References

[1] Bodlaender, H.L., Downey, R.G., Fellows, M.R., Hermelin, J.: On problems without polynomial kernels. J. Comput. Syst. Sci. 75, 423–434 (2009)

[2] Chen, J., Kanj, I.A., Jia, W.: Vertex cover: further observations and further improvements. Journal of Algorithms 41, 280–301 (2001)

[3] Downey, R.G., Fellows, M.R.: Parameterized complexity. Springer, New York (1999)

[4] Fellows, M.R.: The Lost Continent of Polynomial Time: Preprocessing and Kernelization. In: Bodlaender, H.L., Langston, M.A. (eds.) IWPEC 2006. LNCS, vol. 4169, pp. 276–277. Springer, Heidelberg (2006)

[5] Flum, J., Grohe, M.: Parameterized Complexity Theory. Texts in Theoretical Computer Science. An EATCS Series. Springer, Berlin (2006)
[6] Guo, J., Niedermeier, R.: Invitation to data reduction and problem kernelization. SIGACT News 38, 31–45 (2007)
[7] Misra, N., Raman, V., Saurabh, S.: Lower bounds on kernelization. Discrete Optim. 8, 110–128 (2011)
[8] Niedermeier, R.: Invitation to fixed-parameter algorithms. Oxford Lecture Series in Mathematics and its Applications, vol. 31. Oxford University Press, Oxford (2006)