Kernelization methods for fixed-parameter tractability

Fedor V. Fomin and Saket Saurabh

1

Kernelization methods for fixed-parameter tractability

1.1 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 this survey we give an overview of some classical and new techniques in the design of kernelization algorithms.

Preprocessing (data reduction or kernelization) as a strategy of coping with hard problems is universally used in almost every implementation. The history of preprocessing, like applying reduction rules to simplify truth functions, can be traced back to the 1950's [34]. A natural question in this regard is how to measure the quality of preprocessing rules proposed for a specific problem. For a long time the mathematical analysis of polynomial time preprocessing algorithms was neglected. The basic reason for this anomaly was that if we start with an instance I of an NP-hard problem and can show that in polynomial time we can replace this with an equivalent instance I' with $|I'| < |I|$ then that would imply P=NP in classical complexity. The situation changed drastically with advent of parameterized complexity [14]. Combining tools from parameterized complexity and classical complexity it has become possible to

derive upper and lower bounds on the sizes of reduced instances, or so called *kernels*. Importance of preprocessing and the mathematical challenges it poses is beautifully expressed in the following quote by Fellows [16]:

It has become clear, however, that far from being trivial and uninteresting, pre-processing has unexpected practical power for real world input distributions, and is mathematically a much deeper subject than has generally been understood.

Historically, the study of kernelization is rooted in parameterized complexity but it appeared soon that the challenges of kernelization tractability are deeply linked to the classical polynomial time tractability. In the classical computational complexity originated from 1970s, we distinguish between tractable computational problems and intractable. This theory classifies computational problems according to time or space, as a function of the size of the input, required by algorithms to solve them. We can solve tractable problems by efficient algorithms, i.e. algorithms running in time polynomial of the input size, and this is the world of polynomial time computations. We also believe that we cannot solve intractable problems efficiently. Ignoring the structural information about the input and defining intractability according to only the input size, can make some problems appear harder than they are. Parameterized complexity tries to address this issue and measures complexity in terms of the input size and additional parameters. In the parameterized complexity the notion of efficiency, as polynomial time computability, is refined by the notion of fixed parameter tractability. The running time of a fixed parameter tractable algorithm is polynomial in the size of the input but can be exponential in terms of parameters. Surprisingly large number of intractable problems have been shown to exhibit fixed parameter tractable algorithms.

Kernelization algorithms are polynomial time algorithms reducing parameterized problems to equivalent problems while the size of the reduced problem is estimated by some function of the parameter. Thus kernelization can be seen as a refinement of the notion of the classical polynomial time tractability from parameterized perspective. The development of kernelization algorithms demonstrate the importance of the second (and maybe even other) measures and indicate that polynomial time computation is much more powerful than it was suggested before.

In this survey 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 [16] and Guo and Niedermeier [16, 22] 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. [31] 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 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' [10]. 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 [5]. 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 [14].

However, on planar graph p-Dominating Set admits kernel with function $h(k) = \mathcal{O}(k)$, i.e. a linear kernel.

1.2 Basic Definitions

Here we mainly follow notations from the book of Flum and Grohe [18]. 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.

While parameterization $\kappa(x,k) = k$ is the most natural and classical, there can be many other ways of parameterization. For example, take a classical parameterization of the minimum vertex cover problem.

p-Vertex Cover	
Instance:	A graph $G = (V, E)$, and a non-negative integer k.
Parameter:	k.
Problem:	Decide whether G has a vertex cover of size at most k.

But since in every graph the minimum size of a vertex cover is always at least the minimum size of a feedback vertex set, the following parameterization can be interesting as well.

p-Vertex Cover by FVS	
Instance:	A graph G, a feedback vertex set F of size k, and non-negative integer ℓ.
Parameter:	k.
Problem:	Decide whether G has a vertex cover of size at most ℓ.

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 [18]. The classical reference on Parameterized Complexity is the book of Downey and Fellows [14]. For more updated material we refer to books of Flum and Grohe [18] and Niedermeier [32].

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, [18, 32]) *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 chapter we survey some of the known techniques for showing that problems admit polynomial kernels.

We conclude this section with some definitions from graph theory. Let $G = (V, E)$ be a graph. For a vertex v in G, we write $N_G(v)$ to denote the set of v's neighbors in G, and we write $\deg_G(v)$ to denote the *degree* of v, that is, the number of v's neighbors in G. If it is clear from the context which graph is meant, we write $N(v)$ and $\deg(v)$, respectively, for short. A graph $G' = (V', E')$ is a *subgraph* of G if $V' \subseteq V$ and $E' \subseteq E$. The subgraph G' is called an *induced subgraph* of G if $E' = \{\{u, v\} \in E \mid u, v \in V'\}$, in this case, G' is also called the subgraph *induced by* V' and denoted with $G[V']$. A vertex v *dominates* a vertex u if $u \in N(v)$.

1.3 Classical Techniques

In this section we give several examples of techniques to obtain kernels. Some of them are almost trivial and some of them are more involved. We start with the parameterized version of Max3Sat. Our other examples in this Section include a polynomial kernel for d-Hitting Set

using the *Sunflower Lemma*, a kernel for Vertex Cover using *crown decomposition* and an exponential kernel for (Edge) Clique Cover.

Max-3-Sat An r-CNF formula $F = c_1 \wedge \cdots \wedge c_m$ on variable set $V(F)$ is a boolean formula where each clause has size exactly r and each clause is a disjunction of literals. In the optimization version of the problem the task is to find a truth assignment satisfying the maximum number of clauses. The parameterized version of the problem is the following.

p-Max-r-Sat	
Instance:	A r-CNF formula F, and a non-negative integer k.
Parameter:	k
Problem:	Decide whether F has a truth assignment satisfying at least k clauses.

Let (F, k) be an instance of p-Max-3-Sat, i.e. the case of $r = 3$, and let m be the number of clauses in F and n the number of variables. It is well known that in any boolean CNF formula, there is an assignment that satisfies at least half of the clauses (given any assignment that does not satisfy half the clauses, its bitwise complement will). So if the parameter k is less than $m/2$, then there is always an assignment to the variables that satisfies at least k of the clauses. In this case, we reduce the instance to the trivial instance with one clause and the parameter $k = 1$, which is always a YES instance. Otherwise, $m \leq 2k$, and so $n \leq 6k$, and (F, k) itself is the kernel of polynomial size.

***p*-*d*-Hitting Set** Our next example is a kernelization for

p-d-Hitting Set	
Instance:	A family $\mathscr{F}$ of sets over an universe $\mathscr{U}$, each of cardinality d and a positive integer k
Parameter:	k
Problem:	Decide whether there is a subset $U \subseteq \mathscr{U}$ of size at most k such that U contains at least one element from each set in $\mathscr{F}$.

Our kernelization algorithm is based on the following widely used Sunflower Lemma. We first define the terminology used in the statement of the lemma. A *sunflower* with k *petals* and a *core* Y is a collection of sets $S_1, S_2, \ldots, S_k$ such that $S_i \cap S_j = Y$ for all $i \neq j$; the sets $S_i \setminus Y$ are petals and we require none of them to be empty. Note that a family of

pairwise disjoint sets is a sunflower (with an empty core). We need the following classical result of Erdős and Rado [15], see also [18].

Lemma 2 **[Sunflower Lemma]** *Let $\mathscr{F}$ be a family of sets over an universe $\mathscr{U}$ each of cardinality d. If $|\mathscr{F}| > d!(k-1)^d$ then $\mathscr{F}$ contains a sunflower with k petals and such a sunflower can be computed in time polynomial in the size of $\mathscr{F}$ and $\mathscr{U}$.*

Now we are ready to prove the following theorem about kernelization for p-d-Hitting Set

Theorem 1 *d-Hitting Set admits a kernel with $\mathcal{O}(k^d \cdot d!)$ sets and $\mathcal{O}(k^d \cdot d! \cdot d)$ elements.*

Proof The crucial observation is that if $\mathscr{F}$ contains a sunflower $S = \{S_1, \cdots, S_{k+1}\}$ of cardinality $k+1$ then every hitting set H of $\mathscr{F}$ of cardinality k must intersect with the core Y of the sunflower S. Indeed, if H does not intersect C, it should intersect each of the $k+1$ disjoint petals $S_i \setminus C$. Therefore if we let $\mathscr{F}' = \mathscr{F} \setminus (S \cup Y)$, then the instances $(\mathscr{U}, \mathscr{F}, k)$ and $(\mathscr{U}, \mathscr{F}', k)$ are equivalent.

Now we apply the Sunflower Lemma for all $d' \in \{1, \cdots, d\}$ on collections of sets with d' elements by repeatedly replacing sunflowers of size at least $k+1$ with their cores until the number of sets for any fixed $d' \in \{1, \cdots, d\}$ is at most $O(k^{d'} d'!)$. We also remove elements which do not belong to any set. Summing over all d', we obtain that the new family of sets $\mathscr{F}'$ contains $\mathcal{O}(k^d \cdot d!)$ sets. Every set contains at most d elements, and thus the amount of elements in the kernel is $\mathcal{O}(k^d \cdot d! \cdot d)$. □

Crown Decomposition: Vertex Cover Crown decomposition is a general kernelization technique that can be used to obtain kernels for many problems. The technique is based on the classical matching theorems of Kőnig and Hall [25, 29].

Definition 4 *A crown decomposition of a graph G is a partitioning of $V(G)$ as C, H and R, where C and H are nonempty and the partition satisfies the following properties.*

1. *C is an independent set.*
2. *There are no edges between vertices of C and R, i.e. H separates C and R;*
3. *Let E' be the set of edges between vertices of C and H. Then E' contains a matching of size $|H|$.*

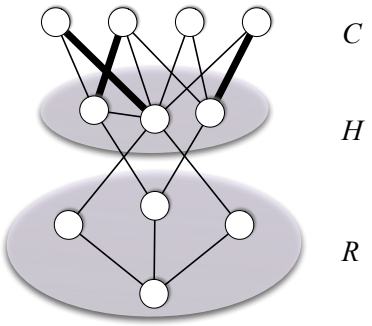

Figure 1.1 Example of a crown decomposition. Set C is an independent set, H separates C and R, and H has a matching into C.

Set C can be seen as a crown put on head H of the remaining part R of the Royal body. Fig. 1.1 provides an example of a crown decomposition. Let us remark that the fact that E' contains a matching of size $|H|$ implies that there is a matching of H into C, i. e. a matching in the bipartite subgraph $G' = (C \cup H, E')$ saturating all the vertices of H.

We demonstrate the application of crown decompositions on kernelization for p-Vertex Cover. p-Vertex Cover can be modelled as p-2-Hitting Set with universe $\mathscr{U} = V$ and $\mathscr{F} = \{\{u, v\} \mid uv \in E\}$ and hence using Theorem 1 we can obtain a kernel with $\mathcal{O}(k^2)$ and $\mathcal{O}(k^2)$ vertices. Here we give a kernel with at most $3k$ vertices.

Given a crown decomposition (C, H, R) of G, one can reduce the instance (G, k) of p-Vertex Cover by making use of the following Rule.

Crown Rule for Vertex Cover: Construct a new instance of the problem (G', k') by removing $H \cup C$ from G and reducing k by $|H|$. In other words, $G' = G[R]$ and $k' = k - |H|$.

Crown Rule is sound by the following lemma.

Lemma 3 *Let (C, H, R) be a crown decomposition of a graph G. Then G has a vertex cover of size k if and only if $G' = G[R]$ has a vertex cover of size $k' = k - |H|$.*

Proof Let S be a vertex cover of G of size k. By the properties of crown decomposition, there is a matching H into C. This matching is of size $|H|$ and it saturates every vertex of H. Thus $|S \cap (H \cup C)| \geq |H|$, as each vertex cover must pick at least one vertex from each of the matching

edge. Hence the number of vertices in S covering the edges not incident to $H \cup C$ is at most $k - |H|$.

For the other direction. If S' is a vertex cover of size $k - |H|$ in G' then $S' \cup H$ is a vertex cover of size k for G. □

Next we show how Crown Rule can be used to obtain a kernel with $3k$ vertices.

Theorem 2 *p-Vertex Cover admits a kernel with at most $3k$ vertices.*

Proof Given an input graph G and a positive integer k, we do as follows. We first find a maximal matching M of G. Let $V(M)$ be the set of endpoints of edges in M. Now if $|V(M)| > 2k$, we answer NO and stop as every vertex cover must contain at least one vertex from each of the matching edges and hence has size more than k.

Let us assume that $|V(M)| \leq 2k$. Because M is a maximal matching, the remaining set $I = V(G) \setminus V(M)$ is an independent set. By Kőnig's Theorem, the minimum size of a vertex cover of a bipartite graph $G_{I,V(M)}$ formed by edges of G between $V(M)$ and I is equal to the maximum size of matching. Thus we can compute a minimum-sized vertex cover of X of this bipartite graph in polynomial time. If $|X| > k$, we again answer NO. So we assume that $|X| \leq k$. If no vertex of X is in $V(M)$, then every vertex of I should be in X (we assume here that graph G has no isolated vertices). But then $|I| \leq k$, and G has at most $3k$ vertices.

So we assume that $X \cap N(I) \neq \emptyset$ but in this case we are able to obtain a crown decomposition (C, H, R) as follows. We put $H = X \cap V(M)$ and $C = I \setminus X$. Obviosely, C is an independent set. Because X is a vertex cover $G_{I,V(M)}$, every vertex of C can be adjacent only to vertices of H. By Kőnig's Theorem, $G_{I,V(M)}$ contains a matching of size $|X|$. Every edge of this matching contains has exactly one endpoint in X, and thus the edges of this matching with endpoints in H form a matching of H into C.

Given a crown decomposition (C, H, R), we apply Crown Rule and obtain a smaller instance for a vertex cover with graph $G' = G[R]$ and parameter $k' = k - |H|$. Now we repeat the above procedure with this reduced instance until either we get a NO answer, or we reach a kernel with $3k$ vertices. □

The bound obtained on the kernel for p-Vertex Cover in Theorem 2 can be further improved to $2k$ with a more sophisticated use of crown

decomposition. An alternate method to obtain a $2k$ size kernel is through a Linear Programming formulation of p-Vertex Cover. See [18] and [32] for further details on Linear Programming based kernelization of p-Vertex Cover.

Clique Cover Unfortunately, not all known problem kernels are shown to have polynomial size. Here, we present some data reduction results with exponential-size kernels. Clearly, it is a pressing challenge to find out whether these bounds can be improved to polynomial ones.

Here is an example

p-Edge Clique Cover	
Instance:	A graph $G = (V, E)$, and a non-negative integer k.
Parameter:	k
Problem:	Decide whether edges of G can be covered by at most k cliques.

We use $N(v)$ to denote the neighborhood of vertex v in G, namely, $N(v) := \{u \mid uv \in E\}$. The *closed* neighborhood of vertex v, denoted by $N[v]$, is $N(v) \cup \{v\}$.

We formulate data reduction rules for a generalized version of p-Edge Clique Cover, in which already some edges may be marked as "covered". Then, the question is to find a clique cover of size k that covers all uncovered edges. We apply the following data reduction rules from [21]:

Rule 1 Remove isolated vertices and vertices that are only adjacent to covered edges

Rule 2 If there is an edge uv whose endpoints have exactly the same closed neighborhood, that is, $N[u] = N[v]$, then mark all edges incident to u as covered. To reconstruct a solution for the non-reduced instance, add u to every clique containing v.

Theorem 3 ([21]) *p-Edge Clique Cover admits a kernel with at most 2^k vertices.*

Proof Let $G = (V, E)$ be a graph that has a clique cover $C_1, \ldots, C_k$ and such that none of two Rules can be applied to G. We claim that G has at most 2^k vertices. Targeting towards a contradiction, let us assume that G has more than 2^k vertices. We assign to each vertex $v \in V$ a binary vector b_v of length k where bit i, $1 \leq i \leq k$, is set to 1 if and only if v is contained in clique C_i. Since there are only 2^k possible

vectors, there must be $u \neq v \in V$ with $b_u = b_v$. If b_u and b_v are zero vectors, the first rule applies; otherwise, u and v are contained in the same cliques. This means that u and v are adjacent and share the same neighborhood, and thus the second rule applies. Hence, if G has more than 2^k vertices, at least one of the reduction rules can be applied to it, which is a contradiction to the initial assumption. □

1.4 Recent Upper Bound Machinery

1.4.1 Protrusion Based Replacement

In this part we discuss the kernelization of different classes of sparse graphs. An important result in the area of kernelization by Alber et al. [1] is a linear sized kernel for the p-Dominating Set problem on planar graphs. This work triggered an explosion of papers on kernelization, and in particular on kernelization of problems on planar and different classes of sparse graphs. Combining the ideas of Alber et al. with *problem specific* data reduction rules, linear kernels were obtained for a variety of parameterized problems on planar graphs including p-Connected Vertex Cover, p-Induced Matching and p-Feedback Vertex Set. In 2009 Bodlaender et al. [4] obtained meta kernelization algorithms that eliminated the need for the design of problem specific reduction rules by providing an automated process that generates them. They show that all problems that have a "distance property" and are expressible in a certain kind of logic or "behave like a regular language" admit a polynomial kernel on graphs of bounded genus. In what follows we give a short description of these meta theorems.

Informal Description. The notion of "protrusions and finite integer index" is central to recent meta kernelization theorems. In the context of problems on graphs, there are three central ideas that form the undercurrent of all protrusion-based reduction rules:

- describing an equivalence that classifies all instances of a problem in an useful manner,
- the ability to easily identify, given a problem, whether the said equivalence has finite index,
- given an instance of a problem, finding large subgraphs that "can be replaced" with smaller subgraphs that are equivalent to the original.

One of the ingenious aspects of this development is coming up with the right definition for describing the circumstances in which a subgraph may be replaced. This is captured by the notion of a protrusion.

In general, an *r-protrusion* in a graph G is simply a subgraph H such that the number of vertices in H that have neighbours in $G \setminus H$ is at most r and the treewidth of H is at most r. (We refer to the chapter of Gottlob and Scarcello "TreeWidth and Hyper-treewidth" for the definition of treewidth.) The *size* of the protrusion is the number of vertices in it, that is, $|V(H)|$. The vertices in H that have neighbours in $G \setminus H$ comprise the *boundary* of H. Informally, H may be thought of as a part of the graph that is separated from the "rest of the graph" by a small-sized separator, and everything about H may be understood in terms of the graph induced by H itself and the limited interaction it has with $G \setminus H$ via its boundary vertices. If the size of the protrusion is large, we may want to replace it with another graph X that is much smaller but which behaviour with respect to $G \setminus H$ is identical to H in the context of the problem that we are studying. Specifically, we would like that the solution to the problem in question does not change after we have made the replacement (or changes in a controlled manner that can be tracked as we make these replacements). This motivates us to define an equivalence that captures the essence of what we hope to do in terms the replacement. We would like to declare H equivalent to X if the size of the solution of G and $(G \setminus H) \cup^* X$ is exactly the same, where $\cup^*$ is some notion of replacement that we have not defined precisely yet. Notice, however, that a natural notion of replacement would leave the boundary vertices intact and perform a cut-and-paste on the rest of H. This is precisely what the protrusion based reduction rules do. Combined with some combinatorial properties of graphs this results in polynomial and, in most cases linear, kernels for variety of problems.

Overview of Meta Kernelization Results. Given a graph $G = (V, E)$, we define $\mathbf{B}_G^r(S)$ to be the set of all vertices of G whose distance from some vertex in S is at most r. Let $\mathcal{G}$ be the family of planar graphs and let integer $k > 0$ be a parameter. We say that a parameterized problem $\Pi \subseteq \mathcal{G} \times \mathbb{N}$ is *compact* if there exist an integer r such that for all $(G = (V, E), k) \in \Pi$, there is a set $S \subseteq V$ such that $|S| \leq r \cdot k$, $\mathbf{B}_G^r(S) = V$ and $k \leq |V|^r$. Similarly, Π is *quasi-compact* if there exists an integer r such that for every $(G, k) \in \Pi$, there is a set $S \subseteq V$ such that $|S| \leq r \cdot k$, $\mathbf{tw}(G \setminus \mathbf{B}_G^r(S)) \leq r$ and $k \leq |V|^r$ where $\mathbf{tw}(G)$ denotes the treewidth of G. Notice that if a problem is compact then it is also

quasi-compact. For ease of presentation the definitions of *compact* and *quasi-compact* are more restrictive here than in the following paper [4].

The following theorem from [4] yields linear kernels for a variety of problems on planar graphs. To this end they utilise the notion of *finite integer index*. This term first appeared in the work by Bodlaender and van Antwerpen-de Fluiter [8] and is similar to the notion of *finite state*. We first define the notion of t-boundaried graphs and the gluing operation. A *t-boundaried graph* is a graph $G = (V, E)$ with t distinguished vertices, uniquely labelled from 1 to t. The set $\partial(G)$ of labelled vertices is called the boundary of G. The vertices in $\partial(G)$ are referred to as boundary vertices or terminals. Let G_1 and G_2 be two t-boundaried graphs. By $G_1 \oplus G_2$ we denote the t-boundaried graph obtained by taking the disjoint union of G_1 and G_2 and identifying each vertex of $\partial(G_1)$ with the vertex of $\partial(G_2)$ with the same label; that is, we *glue* them together on the boundaries. In $G_1 \oplus G_2$ there is an edge between two labelled vertices if there is an edge between them in G_1 or in G_2. For a parameterized problem, Π on graphs in $\mathcal{G}$ and two t-boundaried graphs G_1 and G_2, we say that $G_1 \equiv_\Pi G_2$ if there exists a constant c such that for all t-boundaried graphs G_3 and for all k we have $G_1 \oplus G_3 \in \mathcal{G}$ if and only if $G_2 \oplus G_3 \in \mathcal{G}$ and $(G_1 \oplus G_3, k) \in \Pi$ if and only if $(G_2 \oplus G_3, k+c) \in \Pi$. Note that for every t, the relation $\equiv_\Pi$ on t-boundaried graphs is an equivalence relation. A problem Π has *finite integer index* (FII), if and only if for every t, $\equiv_\Pi$ is of finite index, that is, has a finite number of equivalence classes. Compact problems that have FII include Dominating Set and Connected Vertex Cover while Feedback Vertex Set has FII and is quasi-compact but not compact. We are now in position to state the theorem.

Theorem 4 *Let $\Pi \subseteq \mathcal{G} \times \mathbb{N}$ be quasi-compact and has* FII. *Then Π admits a linear kernel.*

Overview of the Methods. We give an outline of the main ideas used to prove Theorem 4. For a problem Π and an instance $(G = (V, E), k)$ the kernelization algorithm repeatedly identifies a part of the graph to reduce and replaces this part by smaller equivalent part. Since each such step decreases the number of vertices in the graph the process stops after at most $|V|$ iterations. In particular, the algorithm identifies a *constant size separator* S that cuts off a *large* chunk of the graph of *constant treewidth*. This chunk is then considered as a $|S|$-boundaried graph $G' = (V', E')$ with boundary S. Let G^* be the other side of the

separator, that is $G' \oplus G^* = G$. Since Π has FII there exists a finite set $\mathcal{S}$ of $|S|$-boundaried graphs such that $\mathcal{S} \subseteq \mathcal{G}$ and for any $|S|$-boundaried graph G_1 there exists a $G_2 \in \mathcal{S}$ such that $G_2 \equiv_\Pi G_1$. The definition of "large chunk" is that G' should be larger than the largest graph in $\mathcal{S}$. Hence we can find a $|S|$-boundaried graph $G_2 \in \mathcal{S}$ and a constant c such that $(G, k) = (G' \oplus G^*, k) \in \Pi$ if and only if $(G_2 \oplus G^*, k - c) \in \Pi$. The reduction is just to change (G, k) into $(G_2 \oplus G^*, k - c)$. Given G' we can identify G_2 in time linear in $|V'|$ by using the fact that G' has constant treewidth and that all graphs in $\mathcal{S}$ have constant size.

We now proceed to analyze the size of any reduced yes-instance of Π. We show that if Π is compact (not quasi-compact), then the size of a reduced yes-instance (G, k) must be at most $O(k)$. Since $(G = (V, E), k) \in \Pi$ and Π is compact, there is an $O(k)$ sized set $S' \subseteq V$ such that $\mathbf{B}_G^r(S') = V$ for some constant r depending only on Π. One can show that if such a set S' exists there must exist another $O(k)$ sized set S such that the connected components of $G[V \setminus S]$ can be grouped into $O(k)$ chunks as described in the paragraph above. If any of these chunks have more vertices than the largest graph in $\mathcal{S}$ we could have performed the reduction. This implies that any reduced yes-instance has size at most ck for some fixed constant c. Hence if a reduced instance is larger than ck the kernelization algorithm returns NO.

Finally to prove Theorem 4 even when Π is quasi-compact, they show that the set of reduced instances of a quasi-compact problem is in fact compact. Observe that it is the set of reduced instances that becomes compact and *not* Π itself. The main idea is that if $G = (V, E)$ has a set $S \subseteq V$ such that the treewidth of $G[V \setminus \mathbf{B}_G^r(S)]$ is constant and there exists a vertex v which is far away from S, then we can find a large subgraph to reduce.

The parameterized versions of many fundamental optimization problems have finite integer index, including problems like Dominating Set, r-Dominating Set,Vertex Cover, Connected r-Dominating Set, Connected Vertex Cover, Minimum Maximal Matching, Connected Dominating Set, Feedback Vertex Set, Cycle Domination, Edge Dominating Set, Clique Transversal, Independent Set, r-Scattered Set, Min Leaf Spanning Tree, Induced Matching, Triangle Packing, Cycle Packing, Maximum Full-Degree Spanning Tree, and many others [4, 13].

There are problems like Independent Dominating Set, Longest Path, Longest Cycle, Maximum Cut, Minimum Covering by Cliques, Independent Dominating Set, and Minimum Leaf Out-

BRANCHING and various edge packing problems which are known not to have FII [13]. It was shown in [4] that compact problems expressible in an extension of Monadic Second Order Logic, namely Counting Monadic Second Order Logic, have polynomial kernels on planar graphs. This implies polynomial kernels for INDEPENDENT DOMINATING SET, MINIMUM LEAF OUT-BRANCHING, and some edge packing problems on planar graphs. The results from [4] hold not only for planar graphs but for graphs of bounded genus. It was shown in [19] that if instead of quasi-compactness, we request another combinatorial property, bidimensionality with certain separability properties, then an analogue of Theorem 4 can be obtained for much more general graph classes, like graphs excluding some fixed (apex) graph as a minor.

1.4.2 Algebraic and Probabilistic Methods

In Section 1.3, we gave a kernel for p-MAX-3-SAT. Let us discuss in more details kernelization for p-MAX-r-SAT.

Observe that the expected number of clauses satisfied by a random truth assignment that sets each variable of F to one or zero is equal to

$$\mu_F = (1 - 2^{-r})m$$

and thus there is always an assignment satisfying at least μ_F clauses. This implies that at least $m/2$ clauses are always satisfied and hence this parameterization of MAX-r-SAT always has a polynomial kernel because of the following argument. If $k \leq m/2$ then the answer is yes else we have that $m \leq 2k$ and hence $n \leq 2kr$. Thus given a r-CNF formula F, the more meaningful question is whether there exists a truth assignment for F satisfying at least $\mu_F + k$ clauses. We refer to this version of the MAX-r-SAT problem as to p-AG-MAX-r-SAT, that is, the problem where the parameterization is beyond the guaranteed lower bound on the solution.

p-AG-MAX-r-SAT	
Instance:	A r-CNF formula F, and a non-negative integer k.
Parameter:	k
Problem:	Decide whether F has a truth assignment satisfying at least $\mu_F + k$ clauses.

The parameterized study of problems above a guaranteed lower bound was initiated by Mahajan and Raman [30]. They showed that several above guarantee versions of MAX-CUT and MAX-SAT are FPT and

provided a number of open problems around parameterizations beyond guaranteed lower and upper bounds. In a breakthrough paper Gutin et al [23] developed a probabilistic approach to problems parameterized above or below tight bounds. Alon et al. [2] combined this approach with methods from algebraic combinatorics and Fourier analysis to obtain FPT algorithm for parameterized MAX-r-SAT beyond the guaranteed lower bound. Other significant results in this direction include quadratic kernels for ternary permutation constraint satisfaction problems parameterized above average and results around system of linear equations modulo 2 [12, 24]. In what follows we outline the method and an then illustrate the method using an example.

Informal Description of the Method. We give a brief description of the probabilistic method with respect to a given problem Π parameterized above a tight lower bound or below a tight upper bound. We first apply some reductions rules to reduce Π to its special case Π'. Then we introduce a random variable X such that the answer to Π is YES if and only if X takes, with positive probability, a value greater or equal to the parameter k. Now using some probabilistic inequalities on X, we derive upper bounds on the size of NO-instances of Π' in terms of a function of the parameter k. If the size of a given instance exceeds this bound, then we know the answer is YES; otherwise, we produce a problem kernel.

Probabilistic Inequalities. A random variable is *discrete* if its distribution function has a finite or countable number of positive increases. A random variable X is a *symmetric* if $-X$ has the same distribution function as X. If X is discrete, then X is symmetric if and only if $\text{Prob}(X = a) = \text{Prob}(X = -a)$ for each real a. Let X be a symmetric variable for which the first moment $\mathbb{E}(X)$ exists. Then $\mathbb{E}(X) = \mathbb{E}(-X) = -\mathbb{E}(X)$ and, thus, $\mathbb{E}(X) = 0$. The following is easy to prove [23].

Lemma 4 *If X is a symmetric random variable and $\mathbb{E}(X^2) < \infty$, then*

$$\text{Prob}(\ X \geq \sqrt{\mathbb{E}(X^2)}\ ) > 0.$$

Unfortunately, often X is not symmetric, but Lemma 5 provides an inequality that can be used in many such cases. This lemma was proved by Alon et al. [3]; a weaker version was obtained by Håstad and Venkatesh [26].

Lemma 5 *Let X be a random variable and suppose that its first, second and forth moments satisfy $\mathbb{E}(X) = 0$, $\mathbb{E}(X^2) = \sigma^2 > 0$ and $\mathbb{E}(X^4) \leq b\sigma^4$, respectively. Then* $\text{Prob}(\ X > \frac{\sigma}{4\sqrt{b}}\ ) \geq \frac{1}{4^{4/3}b}$.

Since it is often rather nontrivial to evaluate $\mathbb{E}(X^4)$ in order to check whether $\mathbb{E}(X^4) \leq b\sigma^4$ holds, one can sometimes use the following extension of Khinchin's Inequality by Bourgain [9].

Lemma 6 *Let $f = f(x_1, \ldots, x_n)$ be a polynomial of degree r in n variables $x_1, \ldots, x_n$ with domain $\{-1, 1\}$. Define a random variable X by choosing a vector $(\epsilon_1, \ldots, \epsilon_n) \in \{-1, 1\}^n$ uniformly at random and setting $X = f(\epsilon_1, \ldots, \epsilon_n)$. Then, for every $p \geq 2$, there is a constant c_p such that*

$$(\mathbb{E}(|X|^p))^{1/p} \leq (c_p)^r (\mathbb{E}(X^2))^{1/2}.$$

In particular, $c_4 \leq 2^{3/2}$.

An Illustration. Consider the following problem: given a digraph $D = (V, A)$ and a positive integer k, does there exist an acyclic subdigraph of D with at least k arcs? It is easy to prove that this parameterized problem has a linear kernel. Observe that D always has an acyclic subdigraph with at least $|A|/2$ arcs. Indeed, consider a bijection $\alpha : V \to \{1, \ldots, |V|\}$ and the following subdigraphs of D: $(V, \{\, xy \in A : \alpha(x) < \alpha(y) \,\})$ and $(V, \{\, xy \in A : \alpha(x) > \alpha(y) \,\})$. Both subdigraphs are acyclic and at least one of them has at least $|A|/2$ arcs. Thus the input D itself is a kenrel with $2k$ arcs and at most $4k$ vertices. Thus a more natural interesting parameterization is following: decide whether $D = (V, A)$ contains an acyclic subdigraph with at least $|A|/2 + k$ arcs. We choose $|A|/2 + k$ because $|A|/2$ is a *tight lower bound* on the size of a largest acyclic subdigraph. Indeed, the size of a largest acyclic subdigraph of a symmetric digraph $D = (V, A)$ is precisely $|A|/2$. A digraph $D = (V, A)$ is *symmetric* if $xy \in A$ implies $yx \in A$. More precisely we study the following problem.

p-Linear Ordering Above Tight Lower Bound (LOALB)	
Instance:	A digraph D with each arc ij with integer positive weight w_{ij}, and a positive integer k.
Parameter:	k
Problem:	Decide whether is an acyclic subdigraph of D of weight at least $W/2 + k$, where $W = \sum_{ij \in A} w_{ij}$.

Consider the following reduction rule:

Reduction Rule 1 *Assume D has a directed 2-cycle iji; if $w_{ij} = w_{ji}$ delete the cycle, if $w_{ij} > w_{ji}$ delete the arc ji and replace w_{ij} by $w_{ij} - w_{ji}$, and if $w_{ji} > w_{ij}$ delete the arc ij and replace w_{ji} by $w_{ji} - w_{ij}$.*

It is easy to check that the answer to LOALB for a digraph D is YES if and only if the answer to LOALB is YES for a digraph obtained from D using the reduction rule as long as possible.

Let $D = (V, A)$ be an oriented graph, let $n = |V|$ and $W = \sum_{ij \in A} w_{ij}$. Consider a random bijection: $\alpha : V \to \{1, \ldots, n\}$ and a random variable $X(\alpha) = \frac{1}{2} \sum_{ij \in A} \epsilon_{ij}(\alpha)$, where $\epsilon_{ij}(\alpha) = w_{ij}$ if $\alpha(i) < \alpha(j)$ and $\epsilon_{ij}(\alpha) = -w_{ij}$, otherwise. It is easy to see that $X(\alpha) = \sum\{\, w_{ij} : ij \in A, \alpha(i) < \alpha(j) \,\} - W/2$. Thus, the answer to LOALB is YES if and only if there is a bijection $\alpha : V \to \{1, \ldots, n\}$ such that $X(\alpha) \geq k$. Since $\mathbb{E}(\epsilon_{ij}) = 0$, we have $\mathbb{E}(X) = 0$. Let $W^{(2)} = \sum_{ij \in A} w_{ij}^2$. Then one can prove the following:

Lemma 7 ([23]) $\mathbb{E}(X^2) \geq W^{(2)}/12$.

Using Lemma 7 we can show the following. Now we can prove the main result of this section.

Theorem 5 ([23]) *The problem* LOALB *admits a kernel with* $O(k^2)$ *arcs.*

Proof Let H be a digraph. We know that the answer to LOALB for H is YES if and only if the answer to LOALB is YES for a digraph D obtained from H using Reduction Rule 1 as long as possible. Observe that D is an oriented graph. Let $\mathcal{B}$ be the set of bijections from V to $\{1, \ldots, n\}$. Observe that $f : \mathcal{B} \to \mathcal{B}$ such that $f(\alpha(v)) = |V| + 1 - \alpha(v)$ for each $\alpha \in \mathcal{B}$ is a bijection. Note that $X(f(\alpha)) = -X(\alpha)$ for each $\alpha \in \mathcal{B}$. Therefore, $\mathrm{Prob}(X = a) = \mathrm{Prob}(X = -a)$ for each real a and, thus, X is symmetric. Thus, by Lemmas 4 and 7, we have $\mathrm{Prob}(\, X \geq \sqrt{W^{(2)}/12}\,) > 0$. Hence, if $\sqrt{W^{(2)}/12} \geq k$, there is a bijection $\alpha : V \to \{1, \ldots, n\}$ such that $X(\alpha) \geq k$ and, thus, the answer to LOALB (for both D and H) is YES. Otherwise, $|A| \leq W^{(2)} < 12 \cdot k^2$. □

1.5 Conclusion

In this chapter we gave several examples of how parameterized complexity and kernelization allow to analyze different preprocessing algorithms.

In parameterized complexity there there are many reasonable possibilities to "parameterize a problem". For an example for a graph optimization problem a parameter could be the solution size, the structure of the graph (like treewidth or pathwidth), distance of the graph from some polynomially solvable subclasses (for an example deleting at most

k vertices to transform a graph into an interval graph). Other parameters could be obtained by analyzing the hardness proof, or analyzing the data or the dimension. We refer to the survey of Niedermeier [33] for more detailed exposition on this. Bodlaender and Jansen [27] parameterized Vertex Cover by the size of a feedback vertex set. The reason for this parameterization of the vertex cover is interesting because the minimum size of a feedback vertex is always at most the size of the vertex cover number. It was shown in [27] that this parameterized problem admits a cubic kernel. See [6, 7, 27, 28] for other studies of kernelization for parameterization one problem by the solution to the other problem. Parameterizng a graph optimization problem with other graph optimization problem like vertex cover number, max-leaf number have been studied before from the algorithmic perspective [17] but so far there are very few results from the view point of kernelization complexity.

Developing rigorous mathematical theories that explain the behaviour of practical algorithms and heuristics has become an increasingly important challenge in Theory of Computing for the 21st century [11]:

"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."

We believe that kernelization approach based on

(i) identifying parameters that guarantee the tractability, and
(ii) validating that these parameters are low in instances that occur in practice,

can not only shed the light on the mystery of heuristics by providing rigorous mathematical tools to analyze their behaviour but also be a new source of efficient algorithms to handle computational intractability.

References

[1] Jochen Alber, Michael R. Fellows, and Rolf Niedermeier. Polynomial-time data reduction for dominating set. *Journal of the ACM*, 51(3):363–384, 2004.

[2] Noga Alon, Gregory Gutin, Eun Jung Kim, Stefan Szeider, and Anders Yeo. Solving MAX-r-SAT above a tight lower bound. In *Proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2010)*, pages 511–517. SIAM, 2010.

[3] Noga Alon, Gregory Gutin, and Michael Krivelevich. Algorithms with large domination ratio. *J. Algorithms*, 50(1):118–131, 2004.

[4] H. Bodlaender, F. V. Fomin, D. Lokshtanov, E. Penninkx, S. Saurabh, and D. M. Thilikos. (Meta) Kernelization. In *Proceedings of the 50th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2009)*, pages 629–638. IEEE, 2009.

[5] 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.

[6] Hans L. Bodlaender, Bart M. P. Jansen, and Stefan Kratsch. Cross-composition: A new technique for kernelization lower bounds. In *Proceedings of the 28th International Symposium on Theoretical Aspects of Computer Science (STACS 2011)*, volume 9 of *LIPIcs*, pages 165–176. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2011.

[7] Hans L. Bodlaender, Bart M. P. Jansen, and Stefan Kratsch. Preprocessing for treewidth: A combinatorial analysis through kernelization. In *proceedings of the 38th International ColloquiumAutomata, Languages and Programming (ICALP 2011)*, volume 6755 of *Lecture Notes in Comput. Sci.*, pages 437–448. Springer, 2011.

[8] Hans L. Bodlaender and Babette van Antwerpen-de Fluiter. Reduction algorithms for graphs of small treewidth. *Inf. Comput.*, 167(2):86–119, 2001.

[9] J. Bourgain. Walsh subspaces of l^p-product space. *Seminar on Functional Analysis*, Exp. No. 4A, 9, 1980.

[10] Jianer Chen, Iyad A. Kanj, and Weijia Jia. Vertex cover: further obser-

vations and further improvements. *Journal of Algorithms*, 41(2):280–301, 2001.

[11] 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. Available at http://www.cs.buffalo.edu/selman/report/, 1999.

[12] Robert Crowston, Gregory Gutin, Mark Jones, Eun Jung Kim, and Imre Z. Ruzsa. Systems of linear equations over $\mathbb{F}_2$ and problems parameterized above average. In *Proceedings of the 12th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT 2010)*, volume 6139 of *Lecture Notes in Comput. Sci.*, pages 164–175. Springer, 2010.

[13] Babette de Fluiter. *Algorithms for Graphs of Small Treewidth.* PhD thesis, Utrecht University, 1997.

[14] R. G. Downey and M. R. Fellows. *Parameterized complexity.* Springer-Verlag, New York, 1999.

[15] P. Erdős and R. Rado. Intersection theorems for systems of sets. *J. London Math. Soc.*, 35:85–90, 1960.

[16] Michael R. Fellows. The lost continent of polynomial time: Preprocessing and kernelization. In *Proceedings of the 2nd International Workshop on Parameterized and Exact Computation (IWPEC 2006)*, volume 4169 of *Lecture Notes in Comput. Sci.*, pages 276–277. Springer, 2006.

[17] Michael R. Fellows, Daniel Lokshtanov, Neeldhara Misra, Matthias Mnich, Frances A. Rosamond, and Saket Saurabh. The complexity ecology of parameters: An illustration using bounded max leaf number. *Theory Comput. Syst.*, 45(4):822–848, 2009.

[18] Jörg Flum and Martin Grohe. *Parameterized Complexity Theory.* Texts in Theoretical Computer Science. An EATCS Series. Springer-Verlag, Berlin, 2006.

[19] F. V. Fomin, D. Lokshtanov, S. Saurabh, and D. M. Thilikos. Bidimensionality and kernels. In *Proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2010)*, pages 503–510. SIAM, 2010.

[20] Fedor V. Fomin, Daniel Lokshtanov, Neeldhara Misra, Geevarghese Philip, and Saket Saurabh. Hitting forbidden minors: Approximation and kernelization. In *Proceedings of the 28th International Symposium on Theoretical Aspects of Computer Science (STACS 2011)*, volume 9 of *LIPIcs*, pages 189–200. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2011.

[21] Jens Gramm, Jiong Guo, Falk Hüffner, and Rolf Niedermeier. Data reduction and exact algorithms for clique cover. *ACM Journal of Experimental Algorithmics*, 13, 2008.

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

[23] Gregory Gutin, Eun Jung Kim, Stefan Szeider, and Anders Yeo. A probabilistic approach to problems parameterized above or below tight bounds. *J. Comput. Syst. Sci.*, 77(2):422–429, 2011.

[24] Gregory Gutin, Leo van Iersel, Matthias Mnich, and Anders Yeo. All ternary permutation constraint satisfaction problems parameterized above average have kernels with quadratic numbers of variables. In *Proceedings of the 18th Annual European Symposium on Algorithms (ESA 2010)*, volume 6346 of *Lecture Notes in Comput. Sci.*, pages 326–337. Springer, 2010.

[25] Philip Hall. On representatives of subsets. *J. London Math. Soc.*, 10:26–30, 1935.

[26] Johan Håstad and Srinivasan Venkatesh. On the advantage over a random assignment. In *Proceedings of the 34th annual ACM Symposium on Theory of computing (STOC 2002)*, pages 43–52. ACM, 2002.

[27] Bart M. P. Jansen and Hans L. Bodlaender. Vertex cover kernelization revisited: Upper and lower bounds for a refined parameter. In *Proceedings of the 28th International Symposium on Theoretical Aspects of Computer Science (STACS 2011)*, volume 9 of *LIPIcs*, pages 177–188. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2011.

[28] Bart M. P. Jansen and Stefan Kratsch. Data reduction for graph coloring problems. *CoRR*, abs/1104.4229, 2011, to appear in FCT 2011.

[29] Dénes Kőnig. Über Graphen und ihre Anwendung auf Determinantentheorie und Mengenlehre. *Math. Ann.*, 77(4):453–465, 1916.

[30] Meena Mahajan and Venkatesh Raman. Parameterizing above guaranteed values: Maxsat and maxcut. *J. Algorithms*, 31(2):335–354, 1999.

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

[32] Rolf Niedermeier. *Invitation to fixed-parameter algorithms*, volume 31 of *Oxford Lecture Series in Mathematics and its Applications*. Oxford University Press, Oxford, 2006.

[33] Rolf Niedermeier. Reflections on multivariate algorithmics and problem parameterization. In *Proceedings of the 27th International Symposium on Theoretical Aspects of Computer Science (STACS 2010)*, volume 5 of *LIPIcs*, pages 17–32. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2010.

[34] W. V. Quine. The problem of simplifying truth functions. *Amer. Math. Monthly*, 59:521–531, 1952.

[35] Stéphan Thomassé. A quadratic kernel for feedback vertex set. *ACM Transactions on Algorithms*, 6(2), 2010.