Kernelization – Preprocessing with A Guarantee

(For the 60th Birthday of Prof. Mike Fellows)

Daniel Lokshtanov[1] Neeldhara Misra[2] Saket Saurabh[2]

[1] University of California, San Diego, USA
daniello@ii.uib.no
[2] The Institute of Mathematical Sciences, Chennai, India
{neeldhara|saket}@imsc.res.in

Abstract. Data reduction techniques are widely applied to deal with computationally hard problems in real world applications. It has been a long-standing challenge to formally express the efficiency and accuracy of these "pre-processing" procedures. The framework of parameterized complexity turns out to be particularly suitable for a mathematical analysis of pre-processing heuristics. A kernelization algorithm is a preprocessing algorithm which simplifies the instances given as input in polynomial time, and the extent of simplification desired is quantified with the help of the additional parameter.
We give an overview of some of the early work in the area and also survey newer techniques that have emerged in the design and analysis of kernelization algorithms. We also outline the framework of Bodlaender et al. [9] and Fortnow and Santhanam [38] for showing kernelization lower bounds under reasonable assumptions from classical complexity theory, and highlight some of the recent results that strengthen and generalize this framework.

1 Introduction

Preprocessing (data reduction or kernelization) as a strategy of coping with hard problems is universally used in almost every implementation. The history of preprocessing, such as applying reduction rules to simplify truth functions, can be traced back to the 1950's [58]. 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. A possible explanation for this phenomenon is 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. This makes it difficult to design the right definitions of efficient processing within classical complexity. The situation changed drastically with advent of parameterized complexity [26]. 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*. The importance of preprocessing and the mathematical challenges it poses is beautifully expressed in the following quote by Fellows [30]:

It has become clear, however, that far from being trivial and uninteresting, that 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 classical polynomial time tractability. In the classical computational complexity originated from 1970s, we distinguish between tractable computational problems and intractable. This theory classifies problems according to how much time or space is required by algorithms to solve these problems, as a function of the size of the input. Tre tractability border is drawn at polynomial time solvability - a problem which has a polynomial time algorithm is considered tractable, while one that does not is considered intractable. However, ignoring the structural information about the input and defining intractability based only on input size can make some problems appear harder than they really are. Parameterized complexity attempts to address this issue by measuring computational resources such as time and space in terms of input size and additional parameters. In parameterized complexity the central notion of efficiency is "fixed parameter tractability". The notion may be thought of as a generalization of polynomial time to a multivariate setting. 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. A surprisingly large number of intractable problems have been shown to exhibit fixed parameter tractable algorithms. A kernelization algorithm is a polynomial time algorithm reducing instances of parameterized problems to equivalent instances whose size can be upper bounded by a function of the parameter. Thus kernelization can be seen as a refinement of the notion of the classical polynomial time tractability from a 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 previously suggested.

Informally, in parameterized complexity each problem instance comes with a parameter k. As a warm-up, let us consider a few examples of parameterized problems. 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 vertex cover problem, we call it p-Vertex Cover the parameter is an integer k and we ask whether the input graph has a vertex cover of size at most k. We will use $p-$ to emphasise that we are considering a parameterized problem, rather than its classical counterpart. Another problem, p-Longest Path asks whether a given graph contains a path of length at least k. And finally, p-Dominating Set is to decide whether a given graph has a dominating set of size k, that is, 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 a polynomial time algorithm (the degree of polynomial is independent of k), called a *kernelization* 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. If the 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' [17]. 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 [9], even if the input graph G is required to be planar. Our last example, p-Dominating Set admits no kernel unless FPT=W[2] yielding a collapse of several levels in the parameterized complexity hierarchy [26]. However, on planar graph p-Dominating Set admits a kernel of size $h(k) = \mathcal{O}(k)$, i.e. a linear kernel.

In this survey we discuss some of the classical and recent algorithmic techniques for obtaining kernels, and discuss some of the recent developments in deriving lower bounds on the sizes of the 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. Our objective is simply to give a glimpse into the exciting world of kernelization [27, 52, 6, 36, 12, 38, 9, 13, 22, 50, 11, 20]. We refer to the surveys of Fellows [30] and Guo and Niedermeier [30, 41] for further reading on kernelization algorithms. For a more detailed survey of kernelization lower bounds we refer to survey of Misra et al. [55].

2 Basic Definitions

A *parameterized problem* is a language $L \subseteq \Sigma^* \times \mathbb{N}$, where Σ is a finite alphabet and $\mathbb{N}$ is the set of non-negative integers. The second component is called the *parameter* of the problem. The central notion in parameterized complexity is that of *fixed-parameter tractability*, which means given an instance (x, k) of a parameterized language L, deciding whether $(x, k) \in L$ in time $f(k) \cdot p(|x|)$, where f is an arbitrary function of k alone and p is a polynomial function. Such an algorithm is called a fixed-parameter tractable algorithm and we call a problem that admits an algorithm of this kind fixed-parameter tractable (FPT).

We now turn to the formal notion that captures the notion of simplification, which is what most heuristics do when applied to a problem. A *data reduction rule* for a parameterized language L is a function $\phi : \Sigma^* \times \mathbb{N} \to \Sigma^* \times \mathbb{N}$ that maps an instance (x, k) of L to an equivalent instance (x', k') of L such that

1. ϕ is computable in time polynomial in $|x|$ and k;
2. $|x'| \leq |x|$.

Two instances of L are equivalent if $(x, k) \in L$ if and only if $(x', k') \in L$.

In general, a *kernelization algorithm* consists of a finite set of data reduction rules such that by applying the rules to an instance (x, k) (in some specified order) one obtains an instance (x', k') with the property that $|x'| \leq g(k)$ and $k' \leq$

$g(k)$, for some function g only depending on k. Such a "reduced" instance is called a *problem kernel* and $g(k)$ is called the *kernel size*. Formally, this is defined as follows.

Definition 1. [**Kernelization, Kernel**] [9] *A kernelization algorithm for a parameterized problem $\Pi \subseteq \Sigma^* \times \mathbb{N}$ is an algorithm that, given $(x, k) \in \Sigma^* \times \mathbb{N}$, outputs, in time polynomial in $(|x| + k)$, a pair $(x', k') \in \Sigma^* \times \mathbb{N}$ such that: (a) $(x, k) \in \Pi$ if and only if $(x', k') \in \Pi$ and (b) $|x'|, k' \leq g(k)$, where g is some computable function. The output instance x' is called the kernel, and the function g is referred to as the size of the kernel. If $g(k) = k^{O(1)}$, then we say that Π admits a polynomial kernel.*

It is important to mention here that the early definitions of kernelization required that $k' \leq k$. On an intuitive level this makes sense, as the parameter k measures the complexity of the problem — thus the larger the k, the harder the problem. This requirement was subsequently relaxed, notably in the contest of lower bounds. An advantage of the more liberal notion of kernelization is that it is robust with respect to polynomial transformations of the kernel. However, it limits the connection with practical preprocessing. All the kernels obtained in this paper respect the fact that the output parameter is at most the input parameter, that is, $k' \leq k$.

The following lemma tells use that a parameterized problem Π is in FPT if and only if there exists a computable function g such that Π admits a kernel of size $g(k)$.

Lemma 1 ([27]). *If a parameterized problem Q is* FPT *via a computable function then it admits kernelization.*

Proof. Suppose that there is an algorithm deciding if $x \in Q$ in time $f(k)|x|^c$ time for some computable function f and constant c. If $|x| \geq f(k)$, then we run the decision algorithm on the instance in time $f(k)|x|^c \leq |x|^{c+1}$. 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 $|x| < f(k)$, then the kernelization algorithm outputs x. This yields a kernel of size $f(k)$ for the problem. □

However, kernels obtained by this theoretical result are usually of exponential (or even worse) size, while problem-specific data reduction rules often achieve quadratic ($g(k) = O(k^2)$) or even linear-size ($g(k) = O(k)$) kernels. So a natural question for any concrete FPT problem is whether it admits polynomial-time kernelization to a problem kernel that is bounded by a polynomial function of the parameter ($g(k) = O(k^{O(1)})$).

Polynomial kernels form our basic notion of efficient kernelization. For a comprehensive study of fixed-parameter tractability and kernelization, we refer to the books [26, 34, 56] and the surveys [41, 55].

Notations. We conclude this section with some graph-theoretic notations. We follow the style of [24]. 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 $d_G(v)$ 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 $d(v)$, 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 $u \in N(v)$.

3 Classical Techniques explained via Simple Examples

In this section we give several examples of techniques used to obtain kernels, often polynomial kernels. Some of them are almost trivial and some of them are more involved. We start with the parameterized version of Max-3-Sat. Our other examples in this section include a polynomial kernel for p-Feedback Arc Set in Tournaments (p-FAST), d-Hitting Set using the *Sunflower Lemma*, kernels for p-Dual Vertex Coloring and p-Max-SAT using *crown decomposition* and an exponential kernel for p-Edge Clique Cover.

3.1 Max-3-Sat

Let F be a given boolean CNF 3-SAT formula with n variables and m clauses. 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-3-Sat *Instance:* A 3-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 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, $m \leq 2k$. By deleting all variables that do not occur in any clause we obtain that $n \leq 6k$, implying that the input instance is a kernel of polynomial size..

3.2 Kernelization for FAST

In this section we discuss a kernel for p-FAST. A *tournament* is a directed graph T such that for every pair of vertices $v, u \in V(T)$, either uv or vu is an arc of T. A set of arcs A of T is called a *feedback arc set*, if every cycle of T contains an arc from A. In other words, removal of A from T turns it into an acyclic graph.

p-FAST
Instance: A tournament T and a non-negative integer k.
Parameter: k.
Problem: Decide whether T has a feedback arc set of size at most k.

Lemma 2 ([25]). p-FAST *admits a kernel with at most* $k^2 + 2k$ *vertices.*

Proof. The following observation is useful. A graph is acyclic if and only if it is possible to order its vertices in such a way such that for every arc uv, we have $u < v$. Hence a set of arcs A is an inclusion minimal feedback arc set if and only if A is an inclusion minimal set such that reversing directions of all arcs from A results in an acyclic tournament.

In what follows by a *triangle* we mean a directed triangle. We give two simple reduction rules.

Rule 1 If an arc e is contained in at least $k + 1$ triangles, then reverse e and reduce k by 1.

Rule 2 If a vertex v is not contained in any triangle, then delete v from T.

Let us remark that after applying any of the two rules, the resulting graph is again a tournament.

The first rule is sound because if we do not reverse e, we have to reverse at least one arc from each of $k + 1$ triangles containing e. Thus e belongs to every feedback arc set of size at most k.

For the correctness of the second rule. Let X be the set of heads of arcs with tail in v and let Y be the sets of tails of arcs with head in v. Because T is a tournament, X and Y is a partition of $V(T) \setminus \{v\}$. Since v is not a part of any triangle in T, we have that there is no arc from X to Y. Moreover, for any feedback arc set A_1 of tournament $T[X]$ and any feedback arc set A_2 of tournament $T[Y]$, the set $A_1 \cup A_2$ is feedback arc set of T. Thus (T, k) is a YES instance if and only if $(T \setminus v, k)$ is.

Finally we show that any reduced YES instance T has at most $k(k + 2)$ vertices. Let A be a feedback arc set of T of size at most k. For every arc $e \in A$, aside from the two endpoints of e, there are at most k vertices that are contained in a triangle containing e, because otherwise the first rule would have applied. Since every triangle in T contains an arc of A and every vertex of T is in a triangle, we have that T has at most $k(k + 2)$ vertices. □

3.3 *p*-*d*-Hitting Set

Our next example is a kernelization for the p-d-Hitting Set problem, established in [1]. We follow the presentation in [34].

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 $\mathscr{F}$.

The 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 [29], see also [34].

Lemma 3 ([29, 34]). [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 ([34]). *p-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 $\mathcal{O}(k^d \cdot d! \cdot d)$. □

3.4 Kernels via Crown Decomposition

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 [44, 49].

Definition 2. *A crown decomposition of a graph $G = (V, E)$ is a partitioning of V 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 $|H|$.*

Set C can be seen as a crown put on head H of the remaining part R of the Royal body. Fig. 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.

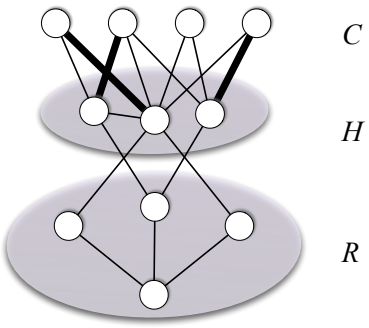

Fig. 1. Example of a crown decomposition. Set C is an independent set, H separates C and R, and H has a matching into C.

The following lemma, which establishes that crown decompositions can be found in polynomial time, is the basis for kernelization algorithms using crown decompositions.

Lemma 4 ([18]). [Crown Lemma] *Let G be a graph without isolated vertices and with at least $3k+1$ vertices. There is a polynomial time algorithm that either*

- *Find a matching of size $k+1$ in G; or*
- *Find a crown decomposition of G.*

We demonstrate the application of crown decompositions on kernelization for p-Dual Vertex Coloring and p-Max-SAT.

Dual of Coloring. In this section we show how Crown Lemma can be used to obtain kernel for p-Dual Vertex Coloring. This problem concerns coloring

of the vertex set of a graph. A k-coloring c of an undirected graph $G = (V, E)$ assigns a color to each vertex of the graph $c : V \to \{1, 2, \ldots, k\}$ such that adjacent vertices have different colours. The smallest k for which G has a k-coloring is called the *chromatic number* of G, denoted by $\chi(G)$. It is well known that deciding if $\chi(G)$ is at most 3 is an NP-complete problem. Thus from the Parameterized Complexity perspective, the following parameterization is more interesting.

p-Dual Vertex Coloring
Instance: A graph $G = (V, E)$ and a non-negative integer k.
Parameter: k
Problem: Decide whether G has a $(\|V\| - k)$-coloring.

It is easier to apply Crown Decomposition to the complement of the input graph. The *complement* of undirected graph $G = (V, E)$ is denoted by $\overline{G}$; its vertex set is V and its edge set is $\overline{E} = \{uv \ : \ uv \notin E, u \neq v\}$. coloring an n-vertex graph in $(n - k)$ colours is equivalent to covering its complement by $(n - k)$ cliques.

Given a crown decomposition (C, H, R) of $\overline{G}$, we apply the following Rule.

Crown Rule for Dual Vertex coloring: 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 $k' = k - |H|$.

The Rule is sound by the following lemma.

Lemma 5 ([18]). *Let (C, H, R) be a crown decomposition of $\overline{G}$. Then $(G = (V, E), k)$ is a* YES *instance if and only if $(G' = (V', E'), k')$ is a* YES *instance.*

Proof. We want to show that G is $(|V| - k)$-colorable if and only if $G' = G[R]$ is $(|V'| - k')$-colorable, where $k' = k - |H|$.

Let c be a $(|V| - k)$-coloring of G. Because C is a clique, all vertices of C are assigned to different colours by c. None of these colours can be used on vertices of R because every vertex from R is adjacent to all vertices of C. Thus c uses on $G' = G[R]$ at most

$$|V| - k - |C| = |V| - (|C| + |H|) - (k - |H|) = |V'| - k'$$

colors.

Now let c' be a $(|V'| - k')$-coloring of G'. We take $|C|$ new colors to color vertices of C. Because there is a matching M of H into C in $\overline{G}$, we can use the same $|C|$ colors that were used on C to color H. For every vertex $u \in H$, we select a color of the vertex from C matched by M to u. To color G, we used at most

$$|V'| - k' + |C| = |V| - (|C| + |H|) - (k - |H|) + |C| = |V| - k$$

colors. This completes the proof. □

Theorem 2 ([18]). *p*-Dual Vertex Coloring *has a kernel with at most* $3k-3$ *vertices.*

Proof. For an input n-vertex graph G and a positive integer k, we take the complement of G. If complement $\overline{G}$ contains an isolated vertex v, then in G this vertex v is adjacent to all other vertices, and thus (G,k) is a YES instance if and only if $(G \setminus v, k-1)$ is a YES instance.

Let us assume that $\overline{G}$ has no isolated vertices. We apply Crown Lemma on $\overline{G}$. If $\overline{G}$ has a matching M of size k, then G is $(n-k)$-colorable. Indeed, the endpoints of every edge of M can be colored with the same color. If $\overline{G}$ has no matching M of size k, then either $n \leq 3(k-1)$, or G can be reduced by making use of the Crown Rule for *p*-Dual Vertex Coloring. □

Maximum Satisfiability. Our next example concerns Max-SAT. We are interested in the following parameterized version of Max-SAT.

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

Theorem 3 ([53]). *p*-Max-SAT *admits a kernel with at most* k *variables and* $2k$ *clauses.*

Proof. Let F be a CNF formula with n variables and m clauses. If we assign values to the variables uniformly at random, linearity of expectation yields that the expected number of satisfied clauses is at least $m/2$. Since there has to be at least one assignment satisfying at least the expected number of clauses this means that if $m \geq 2k$ then (F,k) is a YES instance. In what follows we show how to give a kernel with $n < k$ variables. Whenever possible we apply a cleaning rule; if some variable does not occur in any clauses, remove the variable.

Let G_F be the *variable-clause* incidence graph of F. That is, G_F is a bipartite graph with bipartition (X,Y). The set X corresponds to the variables of F and Y corresponds to the clauses. For a vertex $x \in X$ we will refer to x as both the vertex in G_F and the corresponding variable in F. Similarly, for a vertex $c \in Y$ we will refer to c as both the vertex in G_F and the corresponding clause in F. In G_F there is an edge between a variable $x \in X$ and a clause $c \in Y$ if and only if either x, or its negation is in c. If there is a matching of X into Y in G_F, then there is a truth assignment satisfying at least $|X|$ clauses. This is true because we can set each variable in X in such a way that the clause matched to it becomes satisfied. Thus at least $|X|$ clauses are satisfied. Hence, in this case if $k \leq |X|$ then (F,k) is a YES instance. We now show that if F has at least k variables, then we can in polynomial time, either reduce F to an equivalent smaller instance or find an assignment to the variables satisfying at least k clauses.

Suppose F has at least k variables. Using Hall's Theorem and a polynomial time algorithm computing maximum-size matching, we can in polynomial time find either a matching of X into Y or an inclusion minimal set $C \subseteq X$ such that $|N(C)| < |C|$. If we found a matching we are done, as we can satisfy at least $|X| \geq k$ clauses. So suppose we found a set C as described. Let H be $N(C)$ and $R = V(G_F) \setminus (C \cup H)$. Clearly, $N(C) \subseteq H$, $N(R) \subseteq H$ and $G[C]$ is an independent set. Furthermore, for a vertex $x \in C$ we have that there is a matching of $C \setminus x$ into H since $|N(C')| \geq |C'|$ for every $C' \subseteq C \setminus x$. Since $|C| > |H|$, we have that the matching from $C \setminus x$ to H is in fact a matching of H into C. Hence (C, H, R) is a crown decomposition of G_F.

We prove that all clauses in H are satisfied in every truth assignment to the variables satisfying the maximum number of clauses. Indeed, consider any truth assignment t that does not satisfy all clauses in H. For every variable y in $C \setminus \{x\}$ change the value of y such that the clause in H matched to y is satisfied. Let t' be the new assignment obtained from t in this manner. Since $N(C) \subseteq H$ and t' satisfies all clauses in H, more clauses are satisfied by t' than by t. Hence t can not be an assignment satisfying the maximum number of clauses.

The argument above shows that (F, k) is a YES instance to p-Max-SAT if and only if $(F \setminus (C \cup H), k - |H|)$ is. This gives rise to a simple reduction rule: remove $(C \cup H)$ from F and decrease k by $|H|$. This completes the proof of the theorem. □

3.5 Clique Cover

Unfortunately, not all known problem kernels are shown to have polynomial size. Here, we present the example of p-Edge Clique Cover, and the reduction rules presented here lead to an exponential-size kernel. It has been a pressing challenge for a long time to find out if this can be improved to a polynomial sized kernel. In recent news, the answer to this question has been announced to be in the negative, that is to say, the problem is unlikely to admit a polynomial kernel under reasonable complexity-theoretic assumptions [20]. The problem is the following.

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 describe 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 [40]:

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 4 ([40]). *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. □

4 Recent Upper Bound Machinery

In this section we survey recent methods to obtain polynomial kernels. This includes reduction rules based on protrusions, probabilistic methods and matroids.

4.1 Protrusion Based Replacement

In this part we discuss kernelization for different classes of sparse graphs. An important result in the area of kernelization is by Alber et al. [2]. They obtained 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. [7] 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 critical 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 $H = (V_H, E_H)$ 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. 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 whose 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 a replacement operation 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 [7].

The following theorem from [7] 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 [14] 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 5. *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 5. 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 $\mathbf{B}^r_G(S') = V$ 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 5 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 basic optimization problems have finite integer index, including problems like Dominating Set, (Connected) r-Dominating Set, (Connected) Vertex Cover, Feedback Vertex Set, Edge Dominating Set, Independent Set, Min Leaf Spanning Tree, Induced Matching, Triangle Packing, Cycle Packing, Maximum Full-Degree Spanning Tree, and many others [7, 21].

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 [21]. It was shown in [7] 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 [7] hold not only for planar graphs but for graphs of bounded genus. It was shown in [37] that if instead of quasi-compactness, we request another combinatorial property, bidimensionality with certain separability properties, then an analogue of Theorem 5 can be obtained for much more general graph classes, like graphs excluding some fixed (apex) graph as a minor.

Bodlaender et al. [7] were the first to use protrusion techniques (or rather graph reduction techniques) to obtain kernels, but the idea of using graph replacement for algorithms has been around for long time. The idea of graph replacement for algorithms dates back to Fellows and Langston [31]. Arnborg et al. [6] essentially showed that effective protrusion reduction procedures exist for many problems on graphs of bounded treewidth. Using this, Arnborg et al. [6] obtained a linear time algorithm for MSO expressible problems on graphs of bounded treewidth. Bodlaender and Fluiter [8, 14, 21] generalized these ideas in several ways — in particular, they applied it to some optimization problems. It is also important to mention the work of Bodlaender and Hagerup [10], who used the concept of graph reduction to obtain parallel algorithms for MSO expressible problems on bounded treewidth graphs.

4.2 Algebraic and Probabilistic Methods

A 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 parameterized MAX-r-SAT problem

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.

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 call this version of the MAX-r-SAT problem as p-AG-MAX-r-SAT, that is, 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 [54]. 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 [42] developed a probabilistic approach to problems parameterized above or below tight bounds. Alon et al. [3] 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 [19, 43]. 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 $\mathrm{Prob}(X = a) = \mathrm{Prob}(X = -a)$ 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 [42].

Lemma 6. *If X is a symmetric random variable and $\mathbb{E}(X^2) < \infty$, then*

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

Unfortunately, often X is not symmetric, but Lemma 7 provides an inequality that can be used in many such cases. This lemma was proved by Alon et al. [4]; a weaker version was obtained by Håstad and Venkatesh [45].

Lemma 7. *Let X be a random variable and suppose that its first, second and fourth moments satisfy $\mathbb{E}(X) = 0$, $\mathbb{E}(X^2) = \sigma^2 > 0$ and $\mathbb{E}(X^4) \leq b\sigma^4$, respectively. Then* $\mathrm{Prob}(\ X > \frac{\sigma}{4\sqrt{b}}\ ) \geq \frac{1}{4^{4/3}b}$.

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 [15].

Lemma 8. *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 kernel 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 there 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 9 ([42]). $\mathbb{E}(X^2) \geq W^{(2)}/12$.

Using Lemma 9 we prove the following main result of this section.

Theorem 6 ([42]). *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 6 and 9, 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, $|A| \leq W^{(2)} < 12 \cdot k^2$. □

4.3 Randomized Kernels

A question whether the following problem has a polynomial kernel or not had remained elusive for a few years until recently.

p-Odd Cycle Transversal *Instance:* An undirected graph $G = (V, E)$ and a positive integer k. *Parameter:* k *Problem:* Decide whether there exist a set $S \subseteq V$ such that $G \setminus S$ does not contain odd cycles?

Using techniques from matroid theory it has been recently shown that this problem admits a randomized polynomial kernel [52]. The main part of this kernelization algorithm is to adapt the steps in the FPT algorithm for Odd Cycle Transversal as "independent sets" of the matroid called "gammoid". This exploits the duality between max-flow and min-cut. Recently using another technique from matroid theory a combinatorial kernel has been proposed [51]. This approach works for several other problems including Almost-2-SAT. Even a short description on this algorithm is beyond the scope of this article. We refer the interested readers to the following articles [52, 51].

5 Lower Bound Machinery

Lemma 1 implies that a problem has a kernel if and only if it is fixed parameter tractable. 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? In particular, can we get polynomial sized kernels for problems that admit FPT algorithms? The answer is that quite often we can, as we saw in the previous section, but it turns out that there are a number of problems which are unlikely to have polynomial kernels. It is only very recently that a methodology to rule out polynomial kernels has been developed [9, 38]. The existence of polynomial kernels are ruled out, in this framework, by linking the availability of a polynomial kernel to an unlikely collapse in classical complexity. These developments deepen the connection between classical and parameterized complexity.

In this section we survey the techniques that have been developed to show kernelization lower bounds. To begin with, we consider the following problem.

p-Longest Path *Instance:* An undirected graph $G = (V, E)$ and a non-negative integer k. *Parameter:* k *Problem:* Does G have a path of length k?

It is well known that the p-Longest Path problem can be solved in time $O(c^k n^{O(1)})$ using the well known method of Color-Coding [5]. Is it feasible

that it also admits a polynomial kernel? We argue that intuitively this should not be possible. Consider a large set $(G_1, k), (G_2, k), \ldots, (G_t, k)$ of instances to the p-Longest Path problem. If we make a new graph G by just taking the disjoint union of the graphs $G_1, \ldots, G_t$ we see that G contains a path of length k if and only if G_i contains a path of length k for some $i \leq t$. Suppose the p-Longest Path problem had a polynomial kernel, and we ran the kernelization algorithm on G. Then this algorithm would in polynomial time return a new instance $(G' = (V', E'), k')$ such that $|V'| = k^{O(1)}$, a number potentially much smaller than t. This means that in some sense, the kernelization algorithm considers the instances $(G_1, k), (G_2, k), \ldots, (G_t, k)$ and in *polynomial time* figures out which of the instances are the most likely to contain a path of length k. However, at least intuitively, this seems almost as difficult as solving the instances themselves and since the p-Longest Path problem is NP-complete, this seems unlikely. We now formalize this intuition.

Definition 3. [Distillation [9]]

- *An OR-distillation algorithm for a language $L \subseteq \Sigma^*$ is an algorithm that receives as input a sequence $x_1, \ldots, x_t$, with $x_i \in \Sigma^*$ for each $1 \leq i \leq t$, uses time polynomial in $\sum_{i=1}^{t} |x_i|$, and outputs $y \in \Sigma^*$ with (a) $y \in L \iff x_i \in L$ for* some $1 \leq i \leq t$ *and (b) $|y|$ is polynomial in* $\max_{i \leq t} |x_i|$*. A language L is OR-distillable if there is a OR-distillation algorithm for it.*
- *An AND-distillation algorithm for a language $L \subseteq \Sigma^*$ is an algorithm that receives as input a sequence $x_1, \ldots, x_t$, with $x_i \in \Sigma^*$ for each $1 \leq i \leq t$, uses time polynomial in $\sum_{i=1}^{t} |x_i|$, and outputs $y \in \Sigma^*$ with (a) $y \in L \iff x_i \in L$ for* all $1 \leq i \leq t$ *and (b) $|y|$ is polynomial in* $\max_{i \leq t} |x_i|$*. A language L is AND-distillable if there is an AND-distillation algorithm for it.*

Observe that the notion of distillation is defined for unparameterized problems. Bodlaender et al. [9] conjectured that no NP-complete language can have an OR-distillation or an AND-distillation algorithm.

Conjecture 1. [OR-Distillation Conjecture [9]] No NP-complete language L is OR-distillable.

Conjecture 2. [AND-Distillation Conjecture [9]] No NP-complete language L is AND-distillable.

One should notice that if any NP-complete language is distillable, then so are all of them. Fortnow and Santhanam [38] were able to connect the OR-Distillation Conjecture to a well-known conjecture in classical complexity. In particular they proved that if the OR-Distillation Conjecture fails, then $coNP \subseteq NP/poly$, implying that the *polynomial time hierarchy* [59] collapses to the third level, a collapse that is deemed unlikely. Until very recently, establishing a similar connection for the AND-Distillation Conjecture was one of the central open problems of the area. It is now established that both conjectures hold up to reasonable complexity-theoretic assumptions (see also Section 6.4).

Theorem 7 ([38, 28]).

- *If the OR-Distillation Conjecture fails, then coNP $\subseteq$ NP/poly.*
- *If the AND-Distillation Conjecture fails, then coNP $\subseteq$ NP/poly.*

We are now ready to define the parameterized analogue of distillation algorithms and connect this notion to the Conjectures 1 and 2.

Definition 4. [Composition [9]]

- *A composition algorithm (also called OR-composition algorithm) for a parameterized problem $\Pi \subseteq \Sigma^* \times \mathbb{N}$ is an algorithm that receives as input a sequence $((x_1, k), \ldots, (x_t, k))$, with $(x_i, k) \in \Sigma^* \times \mathbb{N}^+$ for each $1 \leq i \leq t$, uses time polynomial in $\sum_{i=1}^{t} |x_i| + k$, and outputs $(y, k') \in \Sigma^* \times \mathbb{N}^+$ with (a) $(y, k') \in \Pi \iff (x_i, k) \in \Pi$ for* some *$1 \leq i \leq t$ and (b) k' is polynomial in k. A parameterized problem is compositional (or* OR-compositional*) if there is a composition algorithm for it.*
- *An AND-composition algorithm for a parameterized problem $\Pi \subseteq \Sigma^* \times \mathbb{N}$ is an algorithm that receives as input a sequence $((x_1, k), \ldots, (x_t, k))$, with $(x_i, k) \in \Sigma^* \times \mathbb{N}^+$ for each $1 \leq i \leq t$, uses time polynomial in $\sum_{i=1}^{t} |x_i| + k$, and outputs $(y, k') \in \Sigma^* \times \mathbb{N}^+$ with (a) $(y, k') \in \Pi \iff (x_i, k) \in \Pi$ for* all *$1 \leq i \leq t$ and (b) k' is polynomial in k. A parameterized problem is AND-compositional if there is an AND-composition algorithm for it.*

Composition and distillation algorithms are very similar. The main difference between the two notions is that the restriction on output size for distillation algorithms is replaced by a restriction on the parameter size for the instance the composition algorithm outputs. We define the notion of the *unparameterized version* of a parameterized problem L. The mapping of parameterized problems to unparameterized problems is done by mapping (x, k) to the string $x\#1^k$, where $\# \notin \Sigma$ denotes the blank letter and 1 is an arbitrary letter in Σ. In this way, the unparameterized version of a parameterized problem Π is the language $\tilde{\Pi} = \{x\#1^k \mid (x, k) \in \Pi\}$. The following theorem yields the desired connection between the two notions.

Theorem 8 ([9, 28]). *Let Π be a compositional parameterized problem whose unparameterized version $\widetilde{\Pi}$ is* NP*-complete. Then, if Π has a polynomial kernel then coNP $\subseteq$ NP/poly. Similarly, let Π be an AND-compositional parameterized problem whose unparameterized version $\widetilde{\Pi}$ is* NP*-complete. Then, if Π has a polynomial kernel, coNP $\subseteq$ NP/poly.*

We can now formalize the discussion from the beginning of this section.

Theorem 9 ([9]). *p-*Longest Path *does not admit a polynomial kernel unless coNP $\subseteq$ NP/poly.*

Proof. The unparameterized version of *p*-Longest Path is known to be NP-complete [39]. We now give a composition algorithm for the problem. Given

a sequence $(G_1, k), \ldots, (G_t, k)$ of instances we output (G, k) where G is the disjoint union of $G_1, \ldots, G_t$. Clearly G contains a path of length k if and only if G_i contains a path of length k for some $i \leq t$. By Theorem 8 p-Longest Path does not have a polynomial kernel unless $coNP \subseteq NP/poly$. □

An identical proof can be used to show that the p-Longest Cycle problem does not admit a polynomial kernel unless $coNP \subseteq NP/poly$. For many problems, it is easy to give AND-composition algorithms. For instance, the "disjoint union" trick yields AND-composition algorithms for the p-Treewidth, p-Pathwidth and p-Cutwidth problems, among many others. Coupled with Theorem 8 this implies that these problems do not admit polynomial kernels unless $coNP \subseteq NP/poly$.

For some problems, obtaining a composition algorithm directly is a difficult task. Instead, we can give a reduction from a problem that provably has no polynomial kernel unless $coNP \subseteq NP/poly$ to the problem in question such that a polynomial kernel for the problem considered would give a kernel for the problem we reduced from. We now define the notion of *polynomial parameter transformations*.

Definition 5 ([13]). *Let P and Q be parameterized problems. We say that P is polynomial parameter reducible to Q, written $P \leq_{ppt} Q$, if there exists a polynomial time computable function $f : \Sigma^* \times \mathbb{N} \to \Sigma^* \times \mathbb{N}$ and a polynomial p, such that for all $(x, k) \in \Sigma^* \times \mathbb{N}$ (a) $(x, k) \in P$ if and only $(x', k') = f(x, k) \in Q$ and (b) $k' \leq p(k)$. The function f is called* polynomial parameter transformation.

Proposition 1 ([13]). *Let P and Q be the parameterized problems and $\tilde{P}$ and $\tilde{Q}$ be the unparameterized versions of P and Q respectively. Suppose that $\tilde{P}$ is* NP*-complete and $\tilde{Q}$ is in* NP*. Furthermore if there is a polynomial parameter transformation from P to Q, then if Q has a polynomial kernel then P also has a polynomial kernel.*

Proposition 1 shows how to use polynomial parameter transformations to show kernelization lower bounds. A notion similar to polynomial parameter transformation was independently used by Fernau et al. [33] albeit without being explicitly defined. We now give an example of how Proposition 1 can be useful for showing that a problem does not admit a polynomial kernel. In particular, we show that the p-Path Packing problem does not admit a polynomial kernel unless $coNP \subseteq NP/poly$. In this problem you are given a graph G together with an integer k and asked whether there exists a collection of k mutually vertex-disjoint paths of length k in G. This problem is known to be fixed parameter tractable [5] and is easy to see that for this problem the "disjoint union" trick discussed earlier does not directly apply. Thus we resort to polynomial parameter transformations.

Theorem 10. *p-Path Packing does not admit a polynomial kernel unless $coNP \subseteq NP/poly$.*

Proof. We give a polynomial parameter transformation from the p-Longest Path problem. Given an instance (G, k) to p-Longest Path we construct a graph G' from G by adding $k-1$ vertex disjoint paths of length k. Now G contains a path of length k if and only if G' contains k paths of length k. This concludes the proof. □

6 Recent Developments in Lower Bounds

In this section, we provide a brief exposition of some of the more recent developments that have emerged in pursuing lower bounds, namely, cross-compositions, the notion of co-nondeterminism in compositions, and the development that linked the failure of the AND conjecture with an unexpected collapse in classical complexity.

6.1 Cross Composition

Recall that an OR-composition algorithm works by composing multiple instances of a parameterized problem Q into a single instance of $\tilde{Q}$ with a parameter value bounded by a polynomial function of k, the common parameter of all input instances. Further, we also had the constraint that the parameter of the output instance may not depend on the size of the largest input instance, and also should be independent of the number of instances that are input to the algorithm.

It turns out that a variation of the OR-composition algorithm, where the requirements on the output instance are more "relaxed", can still be used to argue lower bounds. This variant was introduced in [11], and is called *cross-composition*. The technique is akin to OR-composition to the extent that it is meant to output the boolean OR of a number of instances. On the other hand, a cross-composition is less restrictive than the standard OR-composition in various ways:

- The source and target problem of the composition need no longer be the same.
- The input to a cross-composition algorithm is a list of classical instances instead of parameterized instances, the inputs do not have a parameter in which the output parameter of the composition must be bounded; instead we require that the size of the output parameter is polynomially bounded in the size of the largest input instance.
- The output parameter may depend polynomially on the logarithm of the number of input instances.

With cross-composition, it is sufficient to compose (via a boolean OR) any classical NP-hard problem into an instance of the parameterized problem Q for which we want to prove a lower-bound, and the parameter of the output instance is permitted to depend on the number of input instances, and the size of the largest instance as well.

For establishing the technique of cross-composition, the notion of a *polynomial equivalence relation* is introduced. Informally, an equivalence relation on Σ^* is a polynomial equivalence relation if it can be "identified" in polynomial time and if the number of equivalence classes of any finite subset are polynomially many in the maximum element of the subset. The formal definition is the following:

Definition 6 (Polynomial equivalence relation, [11]). *An equivalence relation $\mathcal{R}$ on Σ^* is called a* polynomial equivalence relation *if the following two conditions hold:*

1. *There is an algorithm that given two strings $x, y \in \Sigma^*$ decides whether x and y belong to the same equivalence class in $(|x| + |y|)^{O(1)}$ time.*
2. *For any finite set $S \subseteq \Sigma^*$ the equivalence relation $\mathcal{R}$ partitions the elements of S into at most $(\max_{x \in S} |x|)^{O(1)}$ classes.*

We now turn to the definition of cross-composition:

Definition 7 ([11]). *Let $L \subseteq \Sigma^*$ be a set and let $Q \subseteq \Sigma^* \times N$ be a parameterized problem. We say that L* cross-composes into *Q if there is a polynomial equivalence relation $\mathcal{R}$ and an algorithm which, given t strings $x_1, x_2, \ldots, x_t$ belonging to the same equivalence class of $\mathcal{R}$, computes an instance $(x, k) \in \Sigma^* \times N$ in time polynomial in $\sum_{i=1}^{t} |x_i|$ such that:*

1. *$(x, k) \in Q \Rightarrow x_i \in L$ for some $1 \leq i \leq t$,*
2. *k is bounded by a polynomial in $\max_{i=1}^{t} |x_i| + \log t$.*

The existence of a cross-composition from a NP-complete problem into a parameterized problem implies kernel lower bounds for the parameterized problem because a distillation for SAT can be inferred from the cross-composition and the assumption of a polynomial kernel for the parameterized problem. Recall that the existence of a distillation for any NP-complete problem implies that $coNP \subseteq NP/poly$, which completes the argument for the infeasibility of polynomial kernels for problems that admit a cross-composition. Formally, we would say that unless $coNP \subseteq NP/poly$, a problem that admits a cross-composition does not have apolynomial kernel. We now turn to an overview of the argument that leads to a distillation starting from a cross-composition and a polynomial kernel.

Assume that we have a cross-composition from a NP-complete language L to a parameterized language Q. Let m denote the size of the largest input to the distillation algorithm. We describe informally how a cross-composition and a polynomial kernel for Q can be used to devise a distillation algorithm for SAT. For a more formal argument, the reader is referred to [11].

– First, duplicate instances are eliminated from the sequence of inputs to ensure that $t \leq (|\Sigma| + 1)^m$, or that $\log t \in O(m)$. All instances of SAT are transformed into equivalent instances of L (this can be done since L is NP-complete) — note that the sizes of the instances of L are also polynomial in m.

– We now pairwise compare instances using the polynomial-time equivalence test of $\mathcal{R}$ (whose existence is guaranteed by the definition of a cross-composition) to partition the L-instances $(y_1, \ldots, y_t)$ into partite sets $Y_1, \ldots, Y_r$ such that all instances from the same partite set are equivalent under $\mathcal{R}$. The properties of a polynomial equivalence relation guarantee that r is polynomial in m and that this partitioning step takes polynomial time in the total input size.
– Subsequently, a cross-composition is applied to each group of instances in Y_i. In all the parameterized instances that are output by the cross-composition, we have that the parameter is a polynomial function of m, since $\log t \in O(m)$.
– We now apply the kernelization algorithm to obtain polynomial kernels for each instance of Q that is output by the cross-composition. Note that there are polynomially many instances, and each instance after kernelization is also polynomial in size.
– These instances can now be converted back to SAT instances, which can be combined in a straightforward manner to a single instance reflecting the Boolean OR of the original sequence of instances.

Having established what a cross-composition algorithm is, and why it implies kernel lower bounds, we now state some applications of this technique. In [11], the problems considered include p-Chromatic Number and p-Clique parameterized by vertex cover number and p-Feedback Vertex Set parameterized by deletion distance to cluster graphs or co-cluster graphs.

In the case of p-Clique it was already known [9] that the problem does not admit a polynomial kernel parameterized by the treewidth of the graph; since the vertex cover number is at least as large as the treewidth, this is a stronger result. For the unweighted p-Feedback Vertex Set problem, which admits a polynomial kernel parameterized by the target size of the feedback set [16, 60], it can be shown, using cross-composition, that there is no polynomial kernel for the parameterization by deletion distance to cluster graphs or co-cluster graphs.

6.2 Finer Lower Bounds

In [23], the kernel lower bound established by Theorem 8 was generalized further to provide for lower bounds based on different polynomial functions for the kernel size. The OR *of a language* L is the language $\mathrm{OR}(L)$ that consists of all tuples $(x_1, \ldots, x_t)$ for which there is an $i \in \{1, \ldots, t\}$ with $x_i \in L$. Instance $x = (x_1, \ldots, x_t)$ for $\mathrm{OR}(L)$ has two parameters: the length t of the tuple and the maximum bitlength $s = \max_i |x_i|$ of the individual instance for L. The following lemma was established in [23] to prove conditional lower bounds on the kernel sizes.

Lemma 10 ([23]). *Let Π be a problem parameterized by k and let L be an NP-hard problem. Assume that there is a polynomial-time mapping reduction f from $OR(L)$ to Π and a number $d > 0$ with the following property: given an instance $x = (x_1, \ldots, x_t)$ for $OR(L)$ in which each x_i has size at most s, the reduction*

produces an instance $f(x)$ for whose parameter k is at most $t^{\frac{1}{d}+o(1)} \cdot poly(s)$. Then L does not have kernels of size $O(k^{d-\varepsilon})$ for any $\varepsilon > 0$ unless $coNP \subseteq NP/poly$.

Bodlaender et al. [9] formulated this method without the dependency on t. This suffices to prove polynomial kernel lower bounds since d can be chosen as an arbitrarily large constant. It was observed in [23] that the proofs in [9, 38] can be easily adapted to obtain the formulation above, and that it can be generalized to an oracle communication setting. See [23, 22] for more details.

We describe an application of the lemma above to the problem of p-Vertex Cover in graphs, where we have $d = 2$. Following the presentation in [22], we set L to be the p-Multicolored Biclique problem, where the input is a bipartite graph B on the vertex set $U \cup W$, an integer k, and partitions of U and W into k parts, namely $(U_1, \ldots, U_k)$ and $(W_1, \ldots, W_k)$, respectively. We wish to decide if B contains a biclique $K_{k,k}$ that has one vertex from each U_i and W_i for $1 \leq i \leq k$. This is a problem on bipartite graphs and it is NP-complete [22].

Theorem 11 ([23, 22]). *p-Vertex Cover does not have kernels of size $O(k^{2-\varepsilon})$ unless $coNP \subseteq NP/poly$.*

Proof. We apply Lemma 10 where we set L to be p-Multicolored Biclique. Given an instance $(B_1, \ldots, B_t)$ for $OR(L)$, we can assume that every instance B_i has the same number k of groups in the partitions and every group in every instance B_i has the same size n: by simple padding arguments. Furthermore, we can assume that $\sqrt{t}$ is an integer. In the following, we refer to the t instances of p-Multicolored Biclique in the $OR(L)$ instance as $B_{(i,j)}$ for $1 \leq i; j \leq \sqrt{t}$; let $U_{(i,j)}$ and $W_{(i,j)}$ be the two bipartite classes of $B_{(i,j)}$.

First, we modify each instance $B_{(i,j)}$ in such a way that $U_{(i,j)}$ and $W_{(i,j)}$ become complete k-partite graphs: if two vertices $U_{(i,j)}$ or two vertices in $W_{(i,j)}$ are in different groups, then we make them adjacent. It is clear that there is a $2k$-clique in the new graph $B'_{(i,j)}$ if and only if there is a correctly partitioned $K_{k,k}$ in $B_{(i,j)}$. We construct a graph G by introducing $2\sqrt{t}$ sets $(U^1, \ldots, U^{\sqrt{t}})$, $W^1, \ldots, W^{\sqrt{t}}$ of kn vertices each. For every $1 \leq i \leq j \leq \sqrt{t}$, we copy the graph $B'_{(i,j)}$ to the vertex set $U^i \cup W^j$ by mapping $U_{(i,j)}$ to U^i and $W_{(i,j)}$ to W^j. Note that $U_{(i,j)}$ and $W_{(i,j)}$ induces the same complete k-partite graph in $B'_{(i,j)}$ for every i and j, thus this copying can be done in such a way that $G[U^i]$ receives the same set of edges when copying $B'_{(i,j)}$ for any j (and similarly for $G[W^j]$). Therefore, $G[U^i \cup W^j]$ is isomorphic to $B'_{(i,j)}$ for every $1 \leq i \leq j \leq \sqrt{t}$.

It can be verified that G has a $2k$-clique if and only if at least one $B'_{(i,j)}$ has a $2k$-clique (and therefore at least one $B_{(i,j)}$ has a correctly partitioned $K_{k,k}$).

Let $N = 2^{\sqrt{t}}kn$ be the number of vertices in G. Note that $N = t^{1/2} \cdot poly(s)$, where s is the maximum bitlength of the t instances in the $OR(L)$ instance. The graph G has a $2k$-clique if and only if its complement G has a vertex cover of size $N - 2k$. Thus $OR(L)$ can be reduced to an instance of p-Vertex Cover with parameter at most $t^{1/2} \cdot poly(s)$, as required. □

6.3 Co-Nondeterminism in Compositions

In [50], the notion of co-nondeterministic composition is introduced, and it is shown that this concept excludes polynomial kernels, assuming $coNP \not\subseteq NP/poly$. The technique was applied to show that the RAMSEY(k) problem does not admit a polynomial kernel. This is an interesting question posed by Rod Downey — and it asks if the following combination of the well-known CLIQUE and INDEPENDENT SET, known to be NP-complete and FPT, admits a polynomial kernel.

RAMSEY(k)
Instance: An undirected graph G and a non-negative integer k.
Parameter: k.
Problem: Does G contain an independent set or a clique of size k?

Unlike for p-LONGEST PATH [9] (see also Section 5), the disjoint union of t instances of RAMSEY(k) does not work satisfactorily as a composition algorithm (and neither would a join of the instances) as it would contain independent sets of size $\omega(t)$. The intricate Packing Lemma due to Dell and van Melkebeek [23, Lemma 1], although designed in a different context, does not seem to be applicable either as it constructs an n-partite graph containing independent sets of size n which cannot be bounded in $O(\log t)$ when $t := t(n)$ is polynomially-bounded. Generally, it appears to be unlikely that one could pack the instances in such a way that solutions are confined to a part representing a single original instance.

In the context of establishing lower bounds for this problem, the notion of co-nondeterminism in compositions was formulated. A "co-nondeterministic" composition is formally defined as follows:

Definition 8. *Let $Q \subseteq \Sigma^* \times N$. A co-nondeterministic polynomial-time algorithm C is a* coNP composition *for Q if there is a polynomial p such that on input of t instances $(x_1, k), \ldots, (x_t, k) \in \Sigma^* \times N$ the algorithm C takes time polynomial in $\Sigma_{i=1}^{t} |x_i|$ and outputs on each computation path an instance $(y, k') \subseteq \Sigma^* \times N$ with $k' \leq t^{o(1)} p(k)$ and such that the following holds:*

- *If at least one instance (x_i, k) is a yes-instance then all computation paths lead to the output of a yes-instance (y, k').*
- *Otherwise, if all instances (x_i, k) are no-instances, then at least one computation path leads to the output of a no-instance.*

The main tool for establishing that the existence of a coNP composition for a parameterized problem implies a polynomial kernel lower bound for it is a lemma due to Dell and van Melkebeek [23].

It turns out that the combination of a co-nondeterministic composition and a polynomial kernel for a parameterized problem (whose classical version is NP-omplete) implies the existence of an an oracle communication protocol of a suitable kind. This further implies that the corresponding classical problem is in $coNP/poly$, and that finally leads us to the conclusion that $NP \subseteq coNP/poly$, establishing the lower bound.

6.4 The AND Conjecture

As we explained in Section 5, the failing of the AND conjecture did not have any significant implications in classical complexity. If it did have a connection analogous to the one that is enjoyed by the OR conjecture, then this would have several implications in settling the status of the kernelization complexity of a number of problems, to the extent that we make assumptions that are reasonable in the context of classical complexity. For example, in [20], it is shown that p-Edge Clique Cover has no polynomial kernel unless the AND conjecture fails. A number of AND-based-compositions exist for graph layout problems like p-Treewidth, p-Pathwidth, p-Cutwidth and other problems like p-Independent Set, p-Dominating Set when parameterized by the treewidth of the input graph. A more comprehensive discussion can be found in [9]. With this new development, all these problems do not have polynomial kernels unless $NP \subseteq coNP/poly$.

In a talk titled *On the Hardness of Compressing an AND of SAT Instances*, Andrew Drucker revealed that efficient AND-compression would also imply that $NP \subseteq coNP/poly$. To prove this result (and some extensions), any compression scheme is interpreted as a communication channel so as to exploit a certain bottleneck. This entails a new method to "disguise" information being fed into a compressive mapping. At the time of this writing, this work is unpublished, but the details that are available can be found in [28].

7 Conclusion and Discussion

In this section we mention several directions of possible development of kernelization.

Parameterization Vs Parameterizations. In parameterized complexity 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 makes the graph interval). Other parameters could be obtained by analyzing the hardness proof, or analyzing the data or the dimension. We refer to the survey of Niedermeier [57] for more detailed exposition on this. Bodlaender and Jansen [47] parameterized p-Vertex Cover by the size of a feedback vertex set. The reason for this parameterization of the p-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 [47] that this parameterized problem admits a cubic kernel. See [11, 12, 47, 48] for other studies of kernelization for parameterizing 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 [32] but so far there are very few results from the view point of kernelization complexity.

$\mathcal{F}$-Deletion problem. Let $\mathcal{F}$ be a finite set of graphs. In an p-$\mathcal{F}$-Deletion problem, we are given an n-vertex graph G and an integer k as input, and asked whether at most k vertices can be deleted from G such that the resulting graph does not contain a graph from $\mathcal{F}$ as a minor. We refer to such subset as $\mathcal{F}$-hitting set. The p-$\mathcal{F}$-Deletion problem is a generalization of several fundamental problems. For example, when $\mathcal{F} = \{K_2\}$, a complete graph on two vertices, this is p-Vertex Cover. When $\mathcal{F} = \{C_3\}$, a cycle on three vertices, this is the p-Feedback Vertex Set problem. It is known that p-Vertex Cover and p-Feedback Vertex Set admit polynomial kernels [17, 60]. It was shown in [36] that when $\mathcal{F}$ is a graph with two vertices connected by constant number of parallel edges, then p-$\mathcal{F}$-Deletion also admits a polynomial kernel. Recently, it has been shown that p-$\mathcal{F}$-Deletion admits a polynomial kernel whenever $\mathcal{F}$ contains a planar graph [35]. This generalizes several results in the area including for p-Vertex Cover, p-Feedback Vertex Set and p-Pathwidth 1-Deletion. Finally, an interesting direction for further research here is to investigate p-$\mathcal{F}$-Deletion when none of the graphs in $\mathcal{F}$ is planar. The most interesting case here is when $\mathcal{F} = \{K_5, K_{3,3}\}$ aka the Vertex Planarization problem. Surprisingly, we are not aware even of a single case of p-$\mathcal{F}$-Deletion with $\mathcal{F}$ containing no planar graph admitting a polynomial kernel.

Kernelization Lower Bounds. It is known that p-Leaf Out-Branching admits n independent kernels of size $O(k^3)$ [33]. It is not a kernel in the usual "many to one" sense, but it is a kernel in the "one to many" sense. We can generalize the notion of many to one kernels to Turing kernelization. In order to define this we first define the notion of *t-oracle*.

Definition 9. *A t-oracle for a parameterized problem Π is an oracle that takes as input (I, k) with $|I| \leq t$, $k \leq t$ and decides whether $(I, k) \in \Pi$ in constant time.*

Definition 10. *A parameterized problem Π is said to have $g(k)$-sized* turing kernel *if there is an algorithm which given an input (I, k) together with a $g(k)$-oracle for Π decides whether $(I, k) \in \Pi$ in time polynomial in $|I|$ and k. $|x'|, k' \leq g(k)$.*

Observe that both the well known notion of kernels and many to one kernels are special cases of turing kernelization. In particular, many to one kernels are equivalent to turing kernels where the kernelization algorithm is only allowed to make one oracle call and must return the same answer as the oracle.

Problem 1. Is there a framework to rule out the possibility of having one to many or Turing kernels similar to the framework developed in [9, 38]?

Problem 2. Which other problems admit a Turing kernelization like the quadratic kernels for k-Leaf Out-Branching and k-Leaf Out-Tree? Does the problem of finding a path of length at most k admit a Turing kernel (even on planar graphs)?

Problem 3. Does there exist a problem for which we do not have a linear many-to-one kernel, but does have linear kernels from the viewpoint of Turing kernelization?

Recently, there has been an attempt to answer the first question in [46] by organizing problems into complexity classes which are closed under polynomial parameter transformations. It is shown that many of the problems which are known not to have polynomial kernels unless $CoNP \subseteq NP/poly$ are equivalent with respect to Turing kernels. Specifically, either all of them have Turing kernels or all of them do not. The problems belonging to this class include Connected Vertex Cover and Min Ones Sat. Interestingly, Longest Path is not shown to belong to this class, leaving some hope that the problem might have a Turing kernel.

We conclude the survey with the following concrete open problem.

Problem 4. Does p-Directed Feedback Vertex Set admit a polynomial kernel?

References

1. F. N. Abu-Khzam, *A kernelization algorithm for d-hitting set*, J. Comput. Syst. Sci., 76 (2010), pp. 524–531.
2. J. Alber, M. R. Fellows, and R. Niedermeier, *Polynomial-time data reduction for dominating set*, Journal of the ACM, 51 (2004), pp. 363–384.
3. N. Alon, G. Gutin, E. J. Kim, S. Szeider, and A. Yeo, *Solving MAX-r-SAT above a tight lower bound*, in Proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2010), SIAM, 2010, pp. 511–517.
4. N. Alon, G. Gutin, and M. Krivelevich, *Algorithms with large domination ratio*, J. Algorithms, 50 (2004), pp. 118–131.
5. N. Alon, R. Yuster, and U. Zwick, *Color-coding*, J. Assoc. Comput. Mach., 42 (1995), pp. 844–856.
6. S. Arnborg, B. Courcelle, A. Proskurowski, and D. Seese, *An algebraic theory of graph reduction*, J. ACM, 40 (1993), pp. 1134–1164.
7. 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), IEEE, 2009, pp. 629–638.
8. H. L. Bodlaender and B. de Fluiter, *Reduction algorithms for constructing solutions in graphs with small treewidth*, in Proceedings 2nd Annual International Conference on Computing and Combinatorics, COCOON'96, J.-Y. Cai and C. K. Wong, eds., 1996, pp. 199–208.
9. H. L. Bodlaender, R. G. Downey, M. R. Fellows, and D. Hermelin, *On problems without polynomial kernels*, J. Comput. Syst. Sci., 75 (2009), pp. 423–434.
10. H. L. Bodlaender and T. Hagerup, *Parallel algorithms with optimal speedup for bounded treewidth*, SIAM J. Comput., 27 (1998), pp. 1725–1746.
11. H. L. Bodlaender, B. M. P. Jansen, and S. 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), vol. 9 of LIPIcs, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2011, pp. 165–176.

12. ———, *Preprocessing for treewidth: A combinatorial analysis through kernelization*, in proceedings of the 38th International Colloquium Automata, Languages and Programming (ICALP 2011), vol. 6755 of Lecture Notes in Comput. Sci., Springer, 2011, pp. 437–448.
13. H. L. Bodlaender, S. Thomassé, and A. Yeo, *Analysis of data reduction: Transformations give evidence for non-existence of polynomial kernels*, Tech. Rep. CS-UU-2008-030, Department of Information and Computer Sciences, Utrecht University, Utrecht, the Netherlands, 2008.
14. H. L. Bodlaender and B. van Antwerpen-de Fluiter, *Reduction algorithms for graphs of small treewidth*, Inf. Comput., 167 (2001), pp. 86–119.
15. J. Bourgain, *Walsh subspaces of l^p-product space*, Seminar on Functional Analysis, Exp. No. 4A, 9 (1980).
16. K. Burrage, V. Estivill-Castro, M. R. Fellows, M. A. Langston, S. Mac, and F. A. Rosamond, *The undirected feedback vertex set problem has a poly(k) kernel*, in IWPEC, 2006, pp. 192–202.
17. J. Chen, I. A. Kanj, and W. Jia, *Vertex cover: further observations and further improvements*, Journal of Algorithms, 41 (2001), pp. 280–301.
18. B. Chor, M. Fellows, and D. W. Juedes, *Linear kernels in linear time, or how to save k colors in $o(n^2)$ steps*, in Proceedings of the 30th Workshop on Graph-Theoretic Concepts in Computer Science (WG 2004), vol. 3353 of Lecture Notes in Comput. Sci., Springer, 2004, pp. 257–269.
19. R. Crowston, G. Gutin, M. Jones, E. J. Kim, and I. 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), vol. 6139 of Lecture Notes in Comput. Sci., Springer, 2010, pp. 164–175.
20. M. Cygan, S. Kratsch, M. Pilipczuk, M. Pilipczuk, and M. Wahlström, *Clique cover and graph separation: New incompressibility results*, CoRR, abs/1111.0570 (2011).
21. B. de Fluiter, *Algorithms for Graphs of Small Treewidth*, PhD thesis, Utrecht University, 1997.
22. H. Dell and D. Marx, *Kernelization of packing problems*, in SODA, 2012, pp. 68–81.
23. H. Dell and D. van Melkebeek, *Satisfiability allows no nontrivial sparsification unless the polynomial-time hierarchy collapses*, in STOC, 2010, pp. 251–260.
24. R. Diestel, *Graph theory*, vol. 173 of Graduate Texts in Mathematics, Springer-Verlag, Berlin, third ed., 2005.
25. M. Dom, J. Guo, F. Hüffner, R. Niedermeier, and A. Truss, *Fixed-parameter tractability results for feedback set problems in tournaments*, in CIAC, 2006, pp. 320–331.
26. R. G. Downey and M. R. Fellows, *Parameterized complexity*, Springer-Verlag, New York, 1999.
27. R. G. Downey, M. R. Fellows, and U. Stege, *Computational tractability: the view from Mars*, Bull. Eur. Assoc. Theor. Comput. Sci. EATCS, (1999), pp. 73–97.
28. A. Drucker, *On the hardness of compressing an AND of SAT instances*, (2012). Theory Lunch, February 17, Center for Computational Intractability, http://intractability.princeton.edu/blog/2012/03/theory-lunch-february-17/.
29. P. Erdős and R. Rado, *Intersection theorems for systems of sets*, J. London Math. Soc., 35 (1960), pp. 85–90.
30. M. 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), vol. 4169 of Lecture Notes in Comput. Sci., Springer, 2006, pp. 276–277.
31. M. R. Fellows and M. A. Langston, *An analogue of the myhill-nerode theorem and its use in computing finite-basis characterizations (extended abstract)*, in FOCS, 1989, pp. 520–525.
32. M. R. Fellows, D. Lokshtanov, N. Misra, M. Mnich, F. A. Rosamond, and S. Saurabh, *The complexity ecology of parameters: An illustration using bounded max leaf number*, Theory Comput. Syst., 45 (2009), pp. 822–848.
33. H. Fernau, F. V. Fomin, D. Lokshtanov, D. Raible, S. Saurabh, and Y. Villanger, *Kernel(s) for problems with no kernel: On out-trees with many leaves*, in STACS 2009, Schloss Dagstuhl—Leibniz-Zentrum fuer Informatik, 2009, pp. 421–432.
34. J. Flum and M. Grohe, *Parameterized Complexity Theory*, Texts in Theoretical Computer Science. An EATCS Series, Springer-Verlag, Berlin, 2006.
35. F. Fomin, D. Lokshtanov, N. Misra, and S. Saurabh, *Planar-$\mathcal{F}$ Deletion: Approximation, Kernelization and Optimal FPT algorithms*, (2012). Unpublished Manuscript.
36. F. V. Fomin, D. Lokshtanov, N. Misra, G. Philip, and S. Saurabh, *Hitting forbidden minors: Approximation and kernelization*, in Proceedings of the 28th International Symposium on Theoretical Aspects of Computer Science (STACS 2011), vol. 9 of LIPIcs, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2011, pp. 189–200.
37. 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), SIAM, 2010, pp. 503–510.
38. L. Fortnow and R. Santhanam, *Infeasibility of instance compression and succinct PCPs for NP*, in STOC 2008: Proceedings of the 40th Annual ACM Symposium on Theory of Computing, ACM, 2008, pp. 133–142.
39. M. R. Garey and D. S. Johnson, *Computers and Intractability; A Guide to the Theory of NP-Completeness*, W. H. Freeman & Co., New York, NY, USA, 1990.
40. J. Gramm, J. Guo, F. Hüffner, and R. Niedermeier, *Data reduction and exact algorithms for clique cover*, ACM Journal of Experimental Algorithmics, 13 (2008).
41. J. Guo and R. Niedermeier, *Invitation to data reduction and problem kernelization*, SIGACT News, 38 (2007), pp. 31–45.
42. G. Gutin, E. J. Kim, S. Szeider, and A. Yeo, *A probabilistic approach to problems parameterized above or below tight bounds*, J. Comput. Syst. Sci., 77 (2011), pp. 422–429.
43. G. Gutin, L. van Iersel, M. Mnich, and A. 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), vol. 6346 of Lecture Notes in Comput. Sci., Springer, 2010, pp. 326–337.
44. P. Hall, *On representatives of subsets*, J. London Math. Soc., 10 (1935), pp. 26–30.
45. J. Håstad and S. Venkatesh, *On the advantage over a random assignment*, in Proceedings of the 34th annual ACM Symposium on Theory of computing (STOC 2002), ACM, 2002, pp. 43–52.
46. D. Hermelin, S. Kratsch, K. Soltys, M. Wahlström, and X. Wu, *Hierarchies of inefficient kernelizability*, CoRR, abs/1110.0976 (2011).

47. B. M. P. Jansen and H. 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), vol. 9 of LIPIcs, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2011, pp. 177–188.
48. B. M. P. Jansen and S. Kratsch, *Data reduction for graph coloring problems*, in Proceedings of 18th International Symposium on Fundamentals of Computation Theory (FCT 2011), vol. 6914 of Lecture Notes in Comput. Sci., Springer, 2011, pp. 240–251.
49. D. Kőnig, *Über Graphen und ihre Anwendung auf Determinantentheorie und Mengenlehre*, Math. Ann., 77 (1916), pp. 453–465.
50. S. Kratsch, *Co-nondeterminism in compositions: a kernelization lower bound for a ramsey-type problem*, in SODA, 2012, pp. 114–122.
51. S. Kratsch and M. Wahlström, *Representative sets and irrelevant vertices: New tools for kernelization*, CoRR, abs/1111.2195 (2011).
52. ———, *Compression via matroids: a randomized polynomial kernel for odd cycle transversal*, in SODA, 2012, pp. 94–103.
53. D. Lokshtanov, *Phd thesis, New Methods in Parameterized Algorithms and Complexity.*, (2009).
54. M. Mahajan and V. Raman, *Parameterizing above guaranteed values: Maxsat and maxcut*, J. Algorithms, 31 (1999), pp. 335–354.
55. N. Misra, V. Raman, and S. Saurabh, *Lower bounds on kernelization*, Discrete Optim., 8 (2011), pp. 110–128.
56. R. Niedermeier, *Invitation to fixed-parameter algorithms*, vol. 31 of Oxford Lecture Series in Mathematics and its Applications, Oxford University Press, Oxford, 2006.
57. R. Niedermeier, *Reflections on multivariate algorithmics and problem parameterization*, in Proceedings of the 27th International Symposium on Theoretical Aspects of Computer Science (STACS 2010), vol. 5 of LIPIcs, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2010, pp. 17–32.
58. W. V. Quine, *The problem of simplifying truth functions*, Amer. Math. Monthly, 59 (1952), pp. 521–531.
59. L. J. Stockmeyer, *The polynomial-time hierarchy*, Theor. Comp. Sc., 3 (1976), pp. 1–22.
60. S. Thomassé, *A quadratic kernel for feedback vertex set*, ACM Transactions on Algorithms, 6 (2010).