SIGACT News Complexity Theory Column 54

Lane A. Hemaspaandra
Dept. of Computer Science, University of Rochester
Rochester, NY 14627, USA

Introduction to Complexity Theory Column 54

In the June and September issues, this column will feature Fred Green and Steve Homer writing about small depth quantum circuits and Salil Vadhan writing on the complexity of zero knowledge. And warmest thanks to Jiong Guo and Rolf Niedermeier for this issue's article on data reduction and problem kernelization.

Guest Column: Invitation to Data Reduction and Problem Kernelization[1]

Jiong Guo and ***Rolf Niedermeier***

Abstract

To solve NP-hard problems, polynomial-time preprocessing is a natural and promising approach. Preprocessing is based on data reduction techniques that take a problem's input instance and try to perform a reduction to a smaller, equivalent problem kernel. Problem kernelization is a methodology that is rooted in parameterized computational complexity. In this brief survey, we present data reduction and problem kernelization as a promising research field for algorithm and complexity theory.

1 Introduction

There is no denying the fact that preprocessing is a powerful tool in coping with computationally hard problems. Already more than 50 years ago this was observed by Quine [32] when dealing with the simplification of truth functions. For instance, consider the Boolean CNF-Satisfiability problem. To decide whether or not a given formula in conjunctive normal form is satisfiable, one clearly must satisfy all "unit clauses" (consisting of only one literal) by setting the corresponding variable accordingly. Otherwise, there is a choice which may cause a splitting of the search for a satisfying truth assignment by checking the two possible truth values for a variable. In other words, by properly setting variables occurring in unit clauses one may simplify the given formula, avoiding any case distinguishing and related combinatorial explosions. A further well-known simplification rule to deal with CNF-Satisfiability is the pure literal rule: Whenever a variable either occurs only positively or only negatively in a formula, it is sufficient to consider only those truth assignments that set the corresponding literal to true, again avoiding a splitting of the search.

 Institut für Informatik, Friedrich-Schiller-Universität Jena, Ernst-Abbe-Platz 2, D-07743 Jena, Germany. {guo, niedermr}@minet.uni-jena.de. Supported by the Deutsche Forschungsgemeinschaft (DFG), Emmy Noether research group PIAF (fixed-parameter algorithms), NI 369/4.

Apparently, such data reduction rules may not be applicable to every given formula—but as a rule it would be negligent not to try to make use of them whenever possible.

A modern example which shows the striking practical importance of efficient preprocessing is the commercial linear program solver CPLEX. Bixby [5] describes a large linear program that can be solved in half an hour when data reduction is employed but is "far from even being feasible" for the non-reduced instance even after hours of computation.

These two examples shall be enough for a practical motivation of preprocessing and thereby caused data reduction. In this survey we want to briefly describe the state of the art and current challenges in this context. To this end, note that the above described data reductions for CNF-Satisfiability and CPLEX have a strongly heuristic flavor. By way of contrast, our goal here is to set up a formal framework where strong theoretical results can be proven. A way to do so has been shaped by Downey and Fellows with the theory of parameterized complexity analysis [16, 18, 30]. The core idea behind this is given by the problem kernelization concept as we will describe in more detail in a moment. Roughly speaking, the idea is to prove upper bounds on the size of "reduced problem instances". These upper bounds shall be functions solely depending on a parameter that typically is related with the size of a solution set of the problem under study. Although this seems hardly doable for CNF-Satisfiability or CPLEX where the term solution set does not really make sense, for many NP-hard optimization problems we can find such problem parameters. For instance, consider the Vertex Cover problem where one is given an undirected graph and a positive integer k, and the question is whether there is a set C of at most k vertices such that each edge in the graph has at least one endpoint in C. The parameter k reflects the size of the solution set C. And, indeed, one can show by means of data reduction that for every yes-instance of Vertex Cover one can construct an "equivalent" instance with a graph that contains at most $2k$ vertices [29, 12]. This equivalent instance forms the *problem kernel*.

Data reduction and problem kernelization results can explain, and *prove*, why preprocessing works so well in many practical applications; moreover, the quest for kernelization results with provable upper bounds can lead to new and powerful data reduction techniques. We want to emphasize here that experience so far shows that the development of data reduction methods can be entirely nontrivial and surprising, see e.g. Estivill-Castro [17] for making a strong point in this direction. Altogether, we will try to convince the reader that data reduction and problem kernelization is a fascinating field that offers numerous research challenges in algorithm and complexity theory.

2 Basic Concepts and Outline

We study data reduction and problem kernelization within the framework of parameterized complexity theory [16, 18, 30]. To set the stage we need to introduce a few terms and concepts.

We focus our studies on parameterized problems. 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 nonnegative integers. The second component is called the *parameter* of the problem. A parameterized problem is *fixed-parameter tractable* if it can be determined in $f(k) \cdot |x|^{O(1)}$ time whether or not $(x, k) \in L$ where f is a computable function only depending on k. The corresponding complexity class is called FPT. The following two definitions are central to this survey.

Definition 1. *Let $L \subseteq \Sigma^* \times \mathbb{N}$ be a parameterized problem. A* data reduction rule *is a mapping $\phi : \Sigma^* \times \mathbb{N} \to \Sigma^* \times \mathbb{N}$, $(x, k) \mapsto (x', k')$, where ϕ is computable in time polynomial in $|x|$ and k,*

and $(x,k) \in L$ *iff* $(x',k') \in L$ *with* $|x'| \leq |x|$ *and* $k' \leq k$.

Given a finite set $\Phi := \{\phi_1, \phi_2, \ldots, \phi_i\}$ *of data reduction rules,* $\Phi((x,k))$ *denotes the instance after exhaustive application of the data reduction rules in* Φ *to* (x,k) *and is called the* reduced *instance. Then, the term* data reduction process *refers to replacing* (x,k) *by* $\Phi((x,k))$.

It may have remained a bit fuzzy what exhaustive application means. Without getting technical here, it should become clear when we consider the Vertex Cover problem as an example. Before doing that, however, let us present our second central definition.

Definition 2. *Let* L *be a parameterized problem. A* reduction to a problem kernel *or, equivalently,* problem kernelization *means to apply a data reduction process to an instance* (x,k) *such that for the reduced instance* (x',k') *it holds* $|x'| \leq g(k)$ *for some function* g *only depending on* k.

As an example, we consider the NP-complete Vertex Cover problem.

> **Input:** An undirected graph $G = (V,E)$ and a nonnegative integer k.
> **Question:** Is there a set C of at most k vertices such that each edge in E has at least one of its endpoints in C?

We have the following three straightforward data reduction rules. To keep things more readable, we do not use the formal notation as introduced in Definition 1.

1. Remove an isolated vertex and leave $k' := k$.
2. For a degree-one vertex, remove its neighboring vertex from the graph (together with all incident edges) and set $k' := k - 1$.
3. Remove a vertex of degree at least $k+1$ (together with all incident edges) and set $k' := k - 1$.

The first two rules are clearly correct. The third rule is correct because if we did not take v into the cover, then we would have to take every single one of its at least $k+1$ neighbors into the cover in order to cover all edges incident to v. This is not possible because the maximum allowed size of the cover is k. After exhaustively performing the three rules, no vertex in the remaining graph has a degree higher than k, meaning that choosing a vertex into the cover can cause at most k edges to become covered. Since the solution set may be no larger than k, the remaining graph can have at most k^2 edges if it is to have a solution. By the first two rules, every vertex has degree at least two, which implies that the remaining graph can contain at most k^2 vertices. Notably, in the first two rules we did not directly employ the parameter value k—these two rules are *parameter-independent*—whereas in the third rule we did—it is *parameter-dependent*.

There is a one-to-one correspondence between fixed-parameter tractable problems and those problems for which there exists a problem kernel. The proof of the following theorem is not hard.

Theorem 1 ([9]). *Every parameterized problem that is fixed-parameter tractable has a kernelization and vice-versa.*

Unfortunately, the use of this theorem is limited: the running time of a fixed-parameter algorithm directly obtained from a kernelization is usually not practical; and, in the other direction, the theorem does not constructively provide us with a data reduction scheme for a fixed-parameter tractable problem. Thus, the main use of Theorem 1 is to establish the fixed-parameter tractability or amenability to kernelization of a problem—or show that we need not search any further (e.g., if a problem is known to be fixed-parameter intractable, we do not need to look for a kernelization).

Studying computational complexity questions, the smallest size of the problem kernel one can achieve—if a kernelization is doable—is of core interest. Hence, in the next two sections

we first discuss linear-size and polynomial-size problem kernels and second present cases where only exponential-size problem kernels are currently known. These two sections lie at the heart of algorithmic studies of problem kernelization. In terms of computational complexity theory, the derivation of lower bounds is of utmost importance. In addition, aspects of structural complexity theory are interesting when looking at "kernel size classes" (Section 5). In Section 6, we present some final conclusions and challenges for future work. The decision versions of all subsequently studied problems are NP-complete.

3 Linear- and Polynomial-Size Problem Kernels

Here, we present a number of efficient and effective problem kernelizations. The application of most of them may be considered almost as a "must" when trying to solve the corresponding optimization problems. The practical usefulness is undoubted in most cases. Only in the case of Vertex Cover the "final word" about an "ultimate" problem kernel size seems to be spoken. In basically all other cases there is room for improvement—research challenges arise almost everywhere.

3.1 Vertex Cover

Vertex Cover may be considered not only as the drosophila of fixed-parameter algorithmics [30], but also more specifically for the development of data reduction and kernelization techniques. Indeed, the kernel sizes achieved now appear to be in a sense "optimal", as we will discuss later on. The breakthrough for the "ultimate" kernelization for Vertex Cover goes back to an old result of Nemhauser and Trotter [29]. Their result was developed in the context of approximation algorithms but turns out to deliver strong kernelization results. We display two routes to Nemhauser-Trotter kernelization, one based on matching and the other based on linear programming. Refer to Abu-Khzam et al. [1] for a broader exposition.

Kernelization Based on Matching. The subsequently described method guarantees a problem kernel graph where the number of vertices is bounded above by twice the size of an optimal vertex cover of this graph.

Theorem 2 ([29]). *For an n-vertex graph $G = (V, E)$ with m edges, we can compute two disjoint sets $C_0 \subseteq V$ and $V_0 \subseteq V$ in $O(\sqrt{|V|} \cdot |E|)$ time, such that the following three properties hold:*
1. There is a minimum-size vertex cover of G which comprises C_0.
2. The subgraph $G[V_0]$ has a minimum vertex cover of size at least $|V_0|/2$.
3. If $D \subseteq V_0$ is a vertex cover of the induced subgraph $G[V_0]$, then $C := D \cup C_0$ is a vertex cover of G.

Theorem 2 is the key to a problem kernel for Vertex Cover which is formed by a graph that has at least half of its vertices in a minimum-size vertex cover. The algorithm behind Theorem 2 basically consists of a maximum matching computation in a bipartite graph.

Theorem 3 ([29, 12]). Vertex Cover *admits a problem kernel with at most $2k$ vertices.*

Note that the size bound $2k$ for the number of vertices in the reduced graph is the best one may hope for because a problem kernel with $(2 - \varepsilon) \cdot k$ vertices with constant $\varepsilon > 0$ would imply a factor-$(2 - \varepsilon)$ polynomial-time approximation algorithm for Vertex Cover—simply put all

vertices of the reduced graph into the vertex cover which then is by a factor at most $(2-\varepsilon)$ larger than a minimum one. This would be a major breakthrough in approximation theory, since it has been conjectured that it is not possible to polynomial-time approximate Vertex Cover to within a constant factor smaller than 2 [26].

Kernelization Based on Linear Programming. An alternative route to a $2k$-vertices problem kernel for Vertex Cover is rooted in (integer) linear programming techniques, where also the work of Nemhauser and Trotter originated from, and it is based on the following elementary observation. The optimization version of Vertex Cover can be stated as a simple integer linear program: Associate with each graph vertex v a 0/1-variable x_v. Then, $x_v = 1$ means that v is in the vertex cover set and $x_v = 0$ means that it is not. Clearly, for a given graph $G = (V, E)$ Vertex Cover can be expressed as the following optimization problem:

$$\text{Minimize } \textstyle\sum_{v \in V} x_v \text{ where } x_u + x_v \geq 1 \text{ is satisfied for all } \{u, v\} \in E.$$

Since integer linear programming is generally NP-hard, we relax the integer programming formulation to polynomial-time solvable linear programming. We allow x_v to take values in the real-valued interval $0 \leq x_v \leq 1$. The value of the objective function under this relaxation then gives a lower bound for the size of an optimal vertex cover.

Based on the relaxation to linear programming, however, we can obtain a small problem kernel for Vertex Cover. To this end, define

$$C_0 := \{v \in V \mid x_v > 0.5\}, \;\; V_0 := \{v \in V \mid x_v = 0.5\}, \;\; I_0 := \{v \in V \mid x_v < 0.5\}.$$

The following theorem results in a $2k$-vertex problem kernel in the same way as Theorem 2 does.

Theorem 4 ([29]). *Let $(G = (V, E), k)$ be a* Vertex Cover *instance. For C_0, V_0, and I_0 as defined above, there is a minimum-size vertex cover S with $C_0 \subseteq S$ and $S \cap I_0 = \emptyset$. In addition, V_0 induces the problem kernel $(G[V_0], k - |C_0|)$ with $|V_0| \leq 2k$.*

Kernelization Based on Crown Structures. The interest in practically solving Vertex Cover recently led to more direct ways of kernelizing Vertex Cover as we describe next. Recall that vertices of degree one can be easily deleted—we simply can decide to take their neighboring vertices into the vertex cover. This simple observation, in a sense, finds its generalization in the *crown reduction rules*. In this sense, degree-one vertices lead to the most simple crowns.

A *crown* of a graph $G = (V, E)$ consists of $H \subseteq V$ and $I \subseteq V$ with $H \cap I = \emptyset$ such that the following conditions hold:

1. $H = N(I) := (\bigcup_{v \in I} N(v)) \setminus I$,
2. I forms an independent set, and
3. the edges connecting H and I contain a matching in which all elements of H are matched.

It is easy to observe that, given a crown H and I, we can reduce the graph.

Lemma 1 ([13]). *If G is a graph with a crown H and I, then there exists a minimum-size vertex cover of G that contains all of H and none of I.*

Two issues remain to be dealt with, namely how to find crowns efficiently and giving an upper bound on the size of the problem kernel that can be obtained via crown reductions. It turns out that finding crowns can—as with the Nemhauser–Trotter kernelization—be achieved in polynomial time

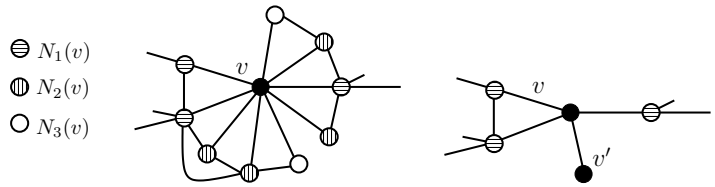

Figure 1: The left-hand side shows the partitioning of the neighborhood of a single vertex v. The right-hand side shows the result of applying the presented data reduction rule to this particular (sub)graph.

by computing maximum matchings [13]. The size of the thus reduced instance is upper-bounded via the following theorem.

Theorem 5 ([13]). *A graph that is crown-free and has a vertex cover of size at most k can contain at most $3k$ vertices.*

Sometimes the $2k$-vertex or $3k$-vertex kernels for Vertex Cover are called linear kernels. Strictly speaking, this is not correct because the instance size is also determined by the number of graph edges, which may be quadratic in the number of graph vertices. The situation changes when considering planar graph problems as we do next.

3.2 Dominating Set in Planar Graphs

Dominating Set in arbitrary graphs is W[2]-complete [16], hence by Theorem 1 there is no hope for a kernelization result in general graphs. For planar graphs, however, the situation looks much better [3]. In Dominating Set in Planar Graphs, we are given a planar graph $G = (V, E)$ and a nonnegative integer k and ask for a set S of at most k vertices such that every vertex $v \in V$ is contained in S or has at least one neighbor in S.

The key to the data reduction for Dominating Set in Planar Graphs are "neighborhood-based" rules. Exemplarily, here we describe a rule where the local neighborhood of single graph vertices is considered. To this end, we need the following definitions.

We partition the neighborhood $N(v)$ of an arbitrary vertex $v \in V$ in the input graph into three disjoint sets $N_1(v)$, $N_2(v)$, and $N_3(v)$ depending on local neighborhood structure. More specifically, we define

1. $N_1(v)$ to contain all neighbors of v that have edges to vertices that are not neighbors of v;
2. $N_2(v)$ to contain all vertices from $N(v) \setminus N_1(v)$ that have edges to at least one vertex from $N_1(v)$;
3. $N_3(v)$ to contain all neighbors of v that are neither in $N_1(v)$ nor in $N_2(v)$.

An example which illustrates such a partitioning is given in Figure 1 (left-hand side).

Now consider a vertex $w \in N_3(v)$. Such a vertex only has neighbors in $\{v\} \cup N_2(v) \cup N_3(v)$. Hence, to dominate w, at least one vertex of $\{v\} \cup N_2(v) \cup N_3(v)$ *must* be contained in a dominating set for the input graph. Since v can dominate all vertices that would be dominated by choosing a vertex from $N_2(v) \cup N_3(v)$ into the dominating set, there is always a minimum dominating set that contains v, and we obtain the following data reduction rule.

> If $N_3(v) \neq \emptyset$ for some vertex v, then remove $N_2(v)$ and $N_3(v)$ from G and add a new vertex v' with the edge $\{v, v'\}$ to G.

Note that the new vertex v' can be considered as a "gadget vertex" that "enforces" v to be chosen into the dominating set, see the right-hand side of Figure 1. It is not hard to verify the correctness of this rule, that is, the original graph has a dominating set of size k iff the reduced graph has a dominating set of size k. Clearly, the data reduction can be executed in polynomial time. Note, however, that there are particular "diamond structures" which are not amenable to this reduction rule. Hence, a second, somewhat more complicated rule based on considering the joint neighborhood of *two* vertices has been introduced [3]. Altogether, the following result could be shown.

Theorem 6 ([3]). Dominating Set in Planar Graphs *admits a problem kernel with $O(k)$ vertices.*

In other words, the theorem states that Dominating Set in Planar Graphs has a linear-size problem kernel. The upper bound on the number of vertices in the problem kernel was first shown to be $335k$ [3] and was then further improved to $67k$ [11]. In addition, methodologically similar results (linear kernelization) have been recently obtained for the Full-Degree Spanning Tree problem in planar graphs [24].

3.3 Maximum Leaf Spanning Tree

So far, we concentrated on minimization problems. Now, we look at the maximization problem Maximum Leaf Spanning Tree, where we are given a graph $G = (V, E)$ and a nonnegative integer k and seek for a spanning tree T of G with at least k leaves.

The following result is particularly interesting because of its use of some form of "extremal graph theory": One knows that every connected graph with minimum degree at least three has a spanning tree with at least $\frac{1}{4}|V| + 2$ leaves [17]. Then, it is easy to give a linear problem kernel for graphs with minimum degree at least three, namely, consisting of $4k$ vertices. Thus, it remains to deal with degree-one and -two vertices.

In the description of the subsequent rules, we often *contract an edge* e. Let $e = \{v, w\}$ and let $N(v)$ and $N(w)$ denote the sets of neighbors of v and w, respectively. Then, contracting e means that we replace v and w by one new vertex x and we set $N(x) := N(v) \cup N(w) \setminus \{v, w\}$. An edge e between two vertices of degree at least two in graph G is called a *bridge* if the removal of e from G results in a graph with one more connected component.

We apply the following data reduction rules due to Estivill-Castro et al. [17]:

1. Contract any bridge.
2. If there is an edge e between two degree-two vertices which is not a bridge, then remove e.
3. If a vertex u has some degree-one neighbors, then remove all these degree-one neighbors and insert edges to transform u's remaining neighbors into a clique. Let c denote the number of the removed degree-one neighbors of u. Decrease the parameter k by $c - 1$.

The correctness of the first two rules is not hard to verify. Concerning the third rule, if a vertex u has more than one degree-one neighbor, then it is safe to remove them except for one and to decrease the parameter by the number of removed vertices. After that, u has only one degree-one neighbor. Then, the rule removes this degree-one vertex and completes u's remaining neighbors to a clique without affecting the parameter. This is correct because in the non-reduced case u cannot be a leaf because of having a degree-one neighbor. But in the reduced case we can always make u into a leaf since u's neighborhood is a clique. Therefore, we can always transform a spanning tree of the non-reduced case into a spanning tree of the reduced case with the same number of leaves

and vice versa. With these three rules one can prove a problem kernel consisting of at most $7.75k$ vertices. A more complicated, extended approach yields the following better bound.

Theorem 7 ([17]). MAXIMUM LEAF SPANNING TREE *admits a problem kernel with at most* $3.75k$ *vertices.*

3.4 Cluster Editing

Our next example is the graph modification problem CLUSTER EDITING which illustrates that clever graph-theoretic concepts may improve a kernel with a quadratic number of vertices [19] to one with a linear number of vertices [21]. In this problem, the question is whether a given graph $G = (V, E)$ can be transformed into a graph that consists of a disjoint union of cliques by deleting or adding at most k edges.

To describe the data reduction rules, we need the following concepts. A *critical clique* of a graph G is a clique K where the vertices of K all have the same set of neighbors in $V \setminus K$, and K is maximal under this property. Let $\mathcal{K}$ be the collection of critical cliques of a graph $G = (V, E)$. Then the *critical clique graph* $\mathcal{C}$ is a graph $(\mathcal{K}, E_{\mathcal{C}})$ with

$$\{K_i, K_j\} \in E_{\mathcal{C}} \iff \forall u \in K_i, v \in K_j : \{u, v\} \in E.$$

That is, the critical clique graph has the critical cliques as nodes, and two nodes are connected iff the corresponding critical cliques together form a larger clique. The critical clique and critical clique graph concepts are similar to the well-known modular decomposition from graph theory.

The basic idea behind introducing critical cliques is the following: suppose that the input graph $G = (V, E)$ has a solution with at most k edge modifications. Then, at most $2k$ vertices are "affected" by these edge modifications, that is, they are endpoints of edges added or deleted. Thus, in order to give a size bound on V depending only on k, it remains to upper-bound the size of the "unaffected" vertices. The central observation is that, in the cluster graph obtained after making the at most k edge modifications, the unaffected vertices contained in one clique must form a critical clique in the original graph G. By this observation, the corresponding data reduction rules [21] upper-bound the maximum size of critical cliques in G by a linear function of k.

Theorem 8 ([21]). CLUSTER EDITING *admits a problem kernel with at most* $4k$ *vertices and* $4k^2 + 4k$ *edges.*

3.5 Feedback Vertex Set

A very recently developed technique for deriving problem kernels is based on approximative solutions, as demonstrated by our next example, FEEDBACK VERTEX SET. Here, we are given an undirected graph $G = (V, E)$ and a nonnegative integer k and ask for a *feedback vertex set* $F \subseteq V$ with $|F| \leq k$ such that each cycle in G contains at least one vertex from F.

It was long open whether there exists a problem kernel of a size polynomial in k for FEEDBACK VERTEX SET. This open problem was recently settled by Burrage et al. [7] who proved an $O(k^{11})$-vertex kernel. After that, Bodlaender [6] improved the kernel size to $O(k^3)$. Both kernelization algorithms use factor-2 approximation algorithms as a first step which compute a feedback vertex set A with $|A| \leq 2k$. Then, this approximative solution serves as a witness structure; based on this witness data reduction rules are applied to reduce the vertices that are not in the approximative solution [6].

Theorem 9 ([6]). Feedback Vertex Set *admits a problem kernel with $O(k^3)$ vertices and $O(k^3)$ edges.*

4 Exponential-Size Problem Kernels

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

4.1 Clique Cover

As the first practically important problem of this section, we study the (Edge) Clique Cover problem, where the input consists of an undirected graph $G = (V, E)$ and a nonnegative integer k and the question is whether there is a set of at most k cliques in G such that each edge in E has both its endpoints in at least one of the selected cliques.

We formulate data reduction rules for a generalized version of 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 [20]:

1. Remove isolated vertices and vertices that are only adjacent to covered edges.
2. If an uncovered edge $\{u, v\}$ is contained in exactly one maximal clique C, that is, if the common neighbors of u and v induce a clique, then add C to the solution, mark its edges as covered, and decrease k by one.
3. If there is an edge $\{u, v\}$ 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.

The correctness of the rules is easy to prove. To show the following problem kernel, only the first and third rule are needed.

Theorem 10 ([20]). Clique Cover *admits a problem kernel with at most 2^k vertices.*

Proof. Consider any graph $G = (V, E)$ with more than 2^k vertices that has a clique cover $C_1, \ldots, C_k$ of size k. 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 iff 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, the first rule applies; otherwise, u and v are contained in the same cliques. This means that u and v are connected and share the same neighborhood, and thus the third rule applies. □

4.2 Further Examples

Multicut in Trees. Multicut is one example of not too many graph problems that remain NP-complete even when restricted to trees. Multicut in Trees has as input an undirected tree $T = (V, E)$, a collection H of m pairs of nodes in V, and a nonnegative integer k. The question is whether there is a set E' of at most k edges such that the removal of the edges in E' disconnects each pair of nodes in H. Using a fairly demanding mathematical analysis, it is possible to show that a reduced instance of Multicut in Trees has a size that can be upper-bounded by $O(k^{3k})$ [22].

Matrix Domination. Matrix Domination is a problem closely related to so-called edge domination problems in graphs. Here, given an $n \times m$-matrix with entries from $\{0, 1\}$ and a nonnegative integer k, we ask for a set C of at most k entries with value 1 such that all other 1-entries are in the same row or column with at least one entry from C. Weston [33] gave three simple data reduction rules and proved a problem kernel of size $O(2^k \cdot k)$.

Capacitated Vertex Cover. Whereas it is easily seen that Vertex Cover allows for a polynomial-size problem kernel, we are far from such a result for the more general Capacitated Vertex Cover problem (CVC), where, given a vertex-weighted and *capacitated* graph G, a nonnegative integer $k \geq 0$, and a nonnegative real number W, one asks for a capacitated vertex cover C for G containing at most k vertices such that $\sum_{c \in C} w(c) \leq W$. "Capacitated" means that each vertex $v \in V$ is assigned an integer *capacity* $c(v) \geq 1$ that limits the number of edges it can cover when being part of the vertex cover. An $O(4^k \cdot k^2)$-vertex problem kernel has been shown in [23].

5 Lower Bounds and Structure Theory

In the previous sections, we have seen several problems possessing linear-size, polynomial-size, or exponential-size problem kernels. It appears natural to substructure the complexity class FPT accordingly. With respect to the exponential combinatorial explosion $f(k)$ in the running time of a fixed-parameter algorithm, first investigations in an analogous way have already been undertaken [18]. Given the practical importance of preprocessing and data reduction, it would be of great interest to find out which structural statements one can make about the (non-)existence of kernels of particular size for various problems. Is there a theory of "non-kernelizability" up to a certain size in analogy to the theory of parameterized intractability that is connected with the W-hierarchy [16, 18, 30]? The only thing one currently knows (cf. Theorem 1) is that parameterized intractable problems do not have problem kernels. But are there any statements of a more fine-grained nature? That is, can we make statements such as "if problem A would have a polynomial-size kernel, then there is some unlikely collapse of some complexity classes"? To our knowledge, this is open.

Clearly, lower bounds are very hard to achieve in computational complexity analysis. Not surprisingly, relatively little is known about lower bounds on kernel sizes. There are two exceptions, though.[2] First, one may make use of known lower bounds on the polynomial-time approximability of certain problems. Recent years have brought several deep results in this direction (PCP theory). Thus, if we know that a certain problem "cannot" be approximated better than a factor c, then we may often conclude that there is no kernel smaller than $c \cdot k$ where parameter k refers to the size of the solution set. More specifically, we can make the following concrete statement about problem kernels for Vertex Cover with parameter k denoting the size of the desired vertex cover:

Theorem 11. Vertex Cover *has no problem kernel formed by a graph with* $1.36k$ *vertices unless* $P{=}NP$.

Proof. Assume that Vertex Cover has a problem kernel with $1.36k$ vertices. Then this set of vertices yields a factor-1.36 polynomial-time approximation of an optimal solution. By deep

[2] Note, however, that the two results we mention are only related to constant factors in "linear" kernel sizes. We are not aware of lower bounds for "non-linear kernels".

results from approximation theory (based on the famous PCP theorem) this is not possible unless P=NP [15]. □

Based on the above approach, we may obtain further lower bounds for problem kernels employing deep results from the theory of polynomial-time approximation. We are not aware of any problem where the lower bound ($1.36k$) and the upper bound ($2k$) on the problem kernel are as close as for Vertex Cover.

Second, we discuss a fairly easy approach, based on elementary calculations and yielding surprising new results. The key observation made use of is based on "parametric duality" [11]. Again, we illustrate matters using Vertex Cover. Observe that an n-vertex graph G has a vertex cover of size k iff G has an independent set of size $|V| - k$. In this sense, the parameters "size of a vertex cover" and "size of an independent set" are duals of each other. Moreover, let us focus attention on planar graphs. The Nemhauser–Trotter problem kernel for Vertex Cover consisting of $2k$ vertices clearly also applies to the special case of planar graphs. In particular, however, due to the four-color theorem and the corresponding polynomial-time coloring algorithm, we also know that there is a trivial $4k$-vertex problem kernel for Independent Set in Planar Graphs. This observation can be used for a "two-sided" algorithmic attack on Vertex Cover in Planar Graphs, yielding the subsequent theorem. Herein, observe that the first approach exhibited with Theorem 11 does not work for Vertex Cover in Planar Graphs. The reason is that there is no constant-ratio lower bound for its polynomial-time approximability because there exists a polynomial-time approximation scheme (PTAS) with approximation ratio $(1 + \epsilon)$ for arbitrarily small $\epsilon > 0$ [4].

Theorem 12 ([11]). *For any $\epsilon > 0$,* Vertex Cover in Planar Graphs *has no problem kernel formed by a planar graph with $(4/3 - \epsilon) \cdot k$ vertices unless P=NP.*

Proof. Consider the polynomial-time kernelization for Vertex Cover's dual problem Independent Set in Planar Graphs—now referred to as "4C"—with upper bound $4k' = 4(|V| - k)$ for the number of vertices in the problem kernel. Assume that there exists a size-$((4/3 - \epsilon) \cdot k)$ problem kernel for Vertex Cover in Planar Graphs. Call the corresponding polynomial-time kernelization algorithm "VK". Let (G, k) be an input instance of Vertex Cover in Planar Graphs and perform the following step.

if $k \leq \frac{4}{4/3-\epsilon+4} \cdot |V|$
then run algorithm VK on (G, k)
else run algorithm 4C on $(G, |V| - k)$
fi

Let $G' = (V', E')$ be the graph of the "reduced instance" computed in the above way. Now, if $k \leq \frac{4}{4/3-\epsilon+4} \cdot |V|$ then

$$|V'| \leq \frac{4 \cdot (4/3 - \epsilon)}{4/3 - \epsilon + 4} \cdot |V| < |V|.$$

Otherwise,

$$|V'| \leq 4 \cdot (|V| - k) < 4 \cdot \left(\frac{4/3 - \epsilon}{4/3 - \epsilon + 4} \right) \cdot |V| < |V|.$$

Thus, in both cases one gets $|V'| < |V|$. Clearly, after a linear number of steps one arrives at a graph with a constant number of vertices—the Vertex Cover problem can be trivially solved

here. Altogether, this would yield a polynomial-time algorithm for the NP-complete Vertex Cover problem, implying P=NP. □

We remark that in the same way one can show that there is no $(2-\epsilon)\cdot k$-vertex problem kernel for Independent Set in Planar Graphs unless P=NP [11]. The above method can be generalized to other problems with "linear size problem kernels" for both the "primal" and the "dual" problem, one further example being Dominating Set in Planar Graphs. The approach seems to be tailored for linear bounds, however; we are not aware of any non-linear lower bound for problem kernels. Much remains to be done in the field for lower bounds for problem kernels.

6 Conclusions and Outlook

In some sense, the concept of problem kernelization can also be viewed as some sort of tightening of the approximation concept. In both cases, we demand efficiency in terms of polynomial processing time. By way of contrast, however, in case of problem kernels we still are asked to guarantee the possibility to find an optimal solution. For instance, recall the $2k$-vertex problem kernel for Vertex Cover due to Nemhauser and Trotter [29]. Just taking all vertices from the kernel into the vertex cover clearly implies an algorithm with approximation factor 2. In this "ideal case", the kernelization even gives the same approximation factor as the best known direct approximation algorithms. Typically, however, the approximations one obtains directly from known kernelization results are relatively weak in comparison with the best approximation results. For instance, the linear-size problem kernel for Dominating Set in Planar Graphs [3, 11] (see Section 3.2) only gives a constant approximation factor whereas the problem is known to have a polynomial-time approximation scheme [4]. This is probably no surprise because, as we pointed out, kernelization—not sacrificing the possibility to find optimal solutions—is demanding and delivering more than approximation does. Still, it is a widely open field to better explore and understand the connections between kernelization and approximation. We state two main topics in this direction.

1. Bodlander's [6] cubic kernel for the Feedback Vertex Set problem (see Section 3.5) employs known factor-2 approximation algorithms. It is a generally interesting field to find out more about how approximation algorithms may help in deriving kernelization results.
2. The other way round, it would be important to learn more about how lower bound results for approximation factors can help to learn more about lower bounds for kernelization. Is there even some analogue to the PCP theory for non-approximability results that allows to show non-kernelizability results?

We only mention in passing that there is also a broader view on the general relationship between parameterized complexity and approximation algorithms; we recommend the recent survey by Marx [28].

Naturally, there are numerous challenges in the young field of kernelizability theory and we can only scratch the surface here. We briefly provide some pointers in the following.

Conceptual Modifications. The kernelization concept as we introduced here may be modified in several ways that all deserve further investigations. We list three such ways to go.

1. Are there, somewhat in analogy to approximation algorithms, reasonable notions of *kernelization schemes*? We see at least two completely routes to follow here. First, there may be some trade-off between smaller kernels versus more involved kernelization algorithms as already indicated

for Dominating Set in Planar Graphs [2]. Second, again looking at planar graphs, one may ask for generally applicable kernelization techniques that work for different problems based on the same strategy. Baker's [4] approximation scheme results should be the model here. Clearly, this issue does not only restrict to planar graph problems.
2. The kernelization framework we presented is based on many-one transformations. Estivill-Castro et al. [17] emphasize that it could be natural and promising to generalize this to a concept of *Turing kernelizability*.
3. In many application scenarios (e.g., bioinformatics) *enumerating* all optimal or sufficiently good solutions is of strong interest. Damaschke [14] reflects on this with the concept of *full kernels* which is worth further investigation.

Concrete Practical Issues. Data reduction and kernelization draws its particular importance from concrete practical application problems [25, 30]. There are several things to do in this context.

It would be valuable to better understand the relationship between *parameter-dependent* and *parameter-independent* data reduction rules. Apparently, parameter-independent rules are more desirable because of their higher flexibility. Similarly, some data reduction rules seem to have a more "*local flavor*" (e.g., the ones for Dominating Set in Planar Graphs)—which seems particularly favorable in distributed implementations—whereas others seem to have a more *global* character (e.g., the Nemhauser-Trotter kernelization). Systematic studies are required here.

It has been observed that in *combination* with search tree algorithms a *repeated application* of kernelization provably pays off [31]. The way how to apply data reduction rules, however, brings up numerous unanswered questions such as in which *order* to apply the rules, whether the order (provably) matters, and how to find the best orders if so.

Given the presumable practical usefulness of data reduction and kernelization, *algorithm engineering* is considered to become a hot topic here.

Impact on Related Fields. As we discussed above, there is a need for better linking data reduction and kernelization with the theory of approximation. There are further important fields:
1. *Average-case analysis* of algorithms is important and challenging. For computationally hard problems, however, it seems natural to examine the average-case behavior *after* a kernelization has been performed.
2. *Parallel and distributed processing* should also be aware of data reduction techniques. Is there a theory of well-parallelizable or well-distributable data reduction techniques?
3. The *compilation concept* [8, 27] that deals with "off-line" preprocessing of parts of input instances demands for investigations about common grounds and differences. Note that Cadoli et al. [8] developed a complexity theory of compilation.

Few Concrete Challenges. We close with three concrete research questions. Note that there are numerous more around!
1. Is there a problem kernel for Feedback Vertex Set with a linear number of vertices?
2. Is there a polynomial-size problem kernel for Clique Cover?
3. Is there a problem kernel for Cluster Editing that implies a constant factor approximation algorithm (cf. [10, 21])?

Acknowledgement. We are grateful to Michael Dom and Hannes Moser for constructive comments on improving our presentation.

References

[1] F. N. Abu-Khzam, R. L. Collins, M. R. Fellows, M. A. Langston, W. H. Suters, and C. T. Symons. Kernelization algorithms for the Vertex Cover problem: Theory and experiments. In *Proc. 6th ACM-SIAM ALENEX*, pages 62–69. ACM-SIAM, 2004.

[2] J. Alber, B. Dorn, and R. Niedermeier. A general data reduction scheme for domination in graphs. In *Proc. 32nd SOFSEM*, volume 3831 of *LNCS*, pages 137–147. Springer, 2006.

[3] J. Alber, M. R. Fellows, and R. Niedermeier. Polynomial time data reduction for Dominating Set. *Journal of the ACM*, 51(3):363–384, 2004.

[4] B. S. Baker. Approximation algorithms for NP-complete problems on planar graphs. *Journal of the ACM*, 41(1):153–180, 1994.

[5] R. E. Bixby. Solving real-world linear programs: A decade and more of progress. *Operations Research*, 50:3–15, 2002.

[6] H. L. Bodlaender. A cubic kernel for feedback vertex set. In *Proc. 24th STACS*, LNCS. Springer, 2007.

[7] K. Burrage, V. Estivill-Castro, M. Fellows, M. Langston, S. Mac, and F. Rosamond. The undirected Feedback Vertex Set problem has a poly(k) kernel. In *Proc. 2nd IWPEC*, volume 4196 of *LNCS*, pages 192–202. Springer, 2006.

[8] M. Cadoli, F. M. Donini, P. Liberatore, and M. Schaerf. Preprocessing of intractable problems. *Information and Computation*, 176(2):89–120, 2002.

[9] L. Cai, J. Chen, R. G. Downey, and M. R. Fellows. Advice classes of parameterized tractability. *Annals of Pure and Applied Logic*, 84:119–138, 1997.

[10] M. Charikar, V. Guruswami, and A. Wirth. Clustering with qualitative information. *Journal of Computer and System Sciences*, 71(3):360–383, 2005.

[11] J. Chen, H. Fernau, I. A. Kanj, and G. Xia. Parametric duality and kernelization: Lower bounds and upper bounds on kernel size. In *Proc. 22nd STACS*, volume 3404 of *LNCS*, pages 269–280. Springer, 2005.

[12] J. Chen, I. A. Kanj, and W. Jia. Vertex Cover: Further observations and further improvements. *Journal of Algorithms*, 41:280–301, 2001.

[13] 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 *Proc. 30th WG*, volume 3353 of *LNCS*, pages 257–269. Springer, 2004.

[14] P. Damaschke. Parameterized enumeration, transversals, and imperfect phylogeny reconstruction. *Theoretical Computer Science*, 351(3):337–350, 2006.

[15] I. Dinur and S. Safra. The importance of being biased. In *Proc. 34th ACM STOC*, pages 33–42. ACM, 2002.

[16] R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Springer, 1999.

[17] V. Estivill-Castro, M. R. Fellows, M. A. Langston, and F. Rosamond. FPT is P-time extremal structure I. In *Proc. 1st ACiD*, volume 4 of *Texts in Algorithmics*, pages 1–41. King's College, 2005.

[18] J. Flum and M. Grohe. *Parameterized Complexity Theory*. Springer, 2006.

[19] J. Gramm, J. Guo, F. Hüffner, and R. Niedermeier. Graph-modeled data clustering: Exact algorithms for clique generation. *Theory of Computing Systems*, 38(4):373–392, 2005.

[20] J. Gramm, J. Guo, F. Hüffner, and R. Niedermeier. Data reduction, exact, and heuristic algorithms for Clique Cover. In *Proc. 8th ACM-SIAM ALENEX*, pages 86–94. ACM-SIAM, 2006. Long version to appear in *The ACM Journal of Experimental Algorithmics*.

[21] J. Guo. A more effective linear kernelization for Cluster Editing, November 2006. Submitted.

[22] J. Guo and R. Niedermeier. Fixed-parameter tractability and data reduction for Multicut in Trees. *Networks*, 46(3):124–135, 2005.

[23] J. Guo, R. Niedermeier, and S. Wernicke. Parameterized complexity of generalized Vertex Cover problems. In *Proc. 9th WADS*, volume 3608 of *LNCS*, pages 36–48. Springer, 2005. Long version to appear under the title "Parameterized complexity of Vertex Cover variants" in *Theory of Computing Systems*.

[24] J. Guo, R. Niedermeier, and S. Wernicke. Fixed-parameter tractability results for full-degree spanning tree and its dual. In *Proc. 2nd IWPEC*, volume 4196 of *LNCS*, pages 203–214. Springer, 2006.

[25] F. Hüffner, R. Niedermeier, and S. Wernicke. Techniques for practical fixed-parameter algorithms. To appear in *The Computer Journal*, 2007.

[26] S. Khot and O. Regev. Vertex Cover might be hard to approximate to within $2-\epsilon$. In *Proc. 18th IEEE Annual Conference on Computational Complexity*, pages 379–386. IEEE, 2003.

[27] P. Liberatore. Monotonic reductions, representative equivalence, and compilation of intractable problems. *Journal of the ACM*, 48(6):1091–1125, 2001.

[28] D. Marx. Parameterized complexity and approximation algorithms. To appear in *The Computer Journal*, 2007.

[29] G. L. Nemhauser and L. E. Trotter. Vertex packing: Structural properties and algorithms. *Mathematical Programming*, 8:232–248, 1975.

[30] R. Niedermeier. *Invitation to Fixed-Parameter Algorithms*. Oxford University Press, 2006.

[31] R. Niedermeier and P. Rossmanith. A general method to speed up fixed-parameter-tractable algorithms. *Information Processing Letters*, 73:125–129, 2000.

[32] W. V. Quine. The problem of simplifying truth functions. *American Mathematical Monthly*, 59:512–531, 1952.

[33] M. Weston. A fixed-parameter tractable algorithm for matrix domination. *Information Processing Letters*, 90:267–272, 2004.