

Kernels: Annotated, Proper and Induced

Faisal N. Abu-Khzam¹ and Henning Fernau²

¹ Lebanese American University, Division of Comp.Sci. & Math., Beirut, Lebanon

² Universität Trier, FB 4—Abteilung Informatik, 54286 Trier, Germany

Abstract. The notion of a “problem kernel” plays a central role in the design of fixed-parameter algorithms. The \mathcal{FPT} literature is rich in kernelization algorithms that exhibit fundamentally different approaches. We highlight these differences and discuss several generalizations and restrictions of the standard notion.

1 Introduction

A *parameterized problem* \mathcal{P} is a usual decision problem together with a special entity called *parameter*. Formally, this means that the language of **YES**-instances of \mathcal{P} , written $L(\mathcal{P})$, is a subset of $\Sigma^* \times \mathbb{N}$. An *instance* of a parameterized problem \mathcal{P} is therefore a pair $(I, k) \in \Sigma^* \times \mathbb{N}$. \mathcal{P} is called *fixed-parameter tractable* if there exists an algorithm for \mathcal{P} running in time $\mathcal{O}(f(k)p(\text{size}(I)))$ on instance (I, k) for some function f and some polynomial p , i.e., the question if $(I, k) \in L(\mathcal{P})$ or not can be decided in time $\mathcal{O}(f(k)p(\text{size}(I)))$. The class of all fixed-parameter tractable parameterized problems is called \mathcal{FPT} . Some other notions of fixed-parameter tractability are also used in the literature (e.g., imposing additional computability constraints), but the above definition is particularly fit for our discussions (especially because of the validity of Theorem 1 below).

The notion of *problem reduction* is at the very core of parameterized complexity. Generally speaking, it can be seen as a particular form of “self-reduction.” However, there seem to be various definitions around that formalize the idea of a self-reduction. A quite general formulation (as can be distilled from [7, p. 39]) seems to be the following one:

Definition 1 (Kernelization). *Let \mathcal{P} be a parameterized problem. A kernelization (reduction) is a function K that is computable in polynomial time and maps an instance (I, k) of \mathcal{P} onto an instance (I', k') of \mathcal{P} such that*

- (I, k) is a **YES**-instance of \mathcal{P} if and only if (I', k') is a **YES**-instance of \mathcal{P}
- $\text{size}(I') \leq f(k)$, and
- $k' \leq g(k)$ for some arbitrary functions f and g .

To underpin the algorithmic nature of a kernelization, K may be referred to as a *kernelization reduction*. (I', k') is also called the (*problem*) *kernel* (of I), and $\text{size}(I')$ the *kernel size*. Of special interest are *polynomial-size kernels* and *linear-size kernels*, where f is a polynomial or a linear function, respectively. A parameterized problem that admits a kernelization is also called *kernelizable*.

Here, $\text{size}(I)$ measures the size of instance I in some “reasonable” manner. In general, it is only the number of bits required to encode I (again, in some “reasonable” manner). More specific size measures are used for example with graphs, where the number of vertices is a further possibility. Observe that the number of bits n' required to store say an adjacency matrix of an n -vertex graph is $n' = \Theta(n^2)$, i.e., there is a polynomial inter-relation between both size measures.

Often, a kernelization function is provided by a set of *kernelization rules* that are to be exhaustively applied to a given instance; we then also speak about a *rule-induced kernelization*.

The notion of a problem kernel is so central to parameterized complexity because of the following well-known result [8]:

Theorem 1. *A parameterized problem is in FPT iff it is kernelizable.*

The proof of Theorem 1 is pretty simple; we indicate the non-trivial direction \Rightarrow : if each problem instance (I, k) is solved in time $f(k)(\text{size}(I))^c$, a reduction running in time $(\text{size}(I))^{c+1}$ can already solve instances with $f(k) \leq \text{size}(I)$. Otherwise, $\text{size}(I) < f(k)$, and then (I, k) can be seen as a “reduced instance.”

However, the given definition of kernelization is not appropriate for several applications. We will discuss the following problems and according modifications of the definition in this paper:

- (a) When proving lower bounds, the notion presented so far seems to be too strong. A slightly weaker version is more convenient to establish a simple relation between hardness of approximation results and lower bounds on problem kernels.
- (b) Some “kernels” as presented in the literature do not fit into our definition, since rather a reduction to a more general problem instance is provided.
- (c) Finally, some problems can be easily solved by search tree techniques through a reduction to a “master problem” like HITTING SET. However, in general the kernel result known for the master problem does not provide an immediate kernel result for the related problem one is interested in, since some ingredients of the master problem kernelization may not have counterparts in the related problem. Thus, it is interesting to investigate *induced kernels* that allow for transferring kernelization rules to a new problem.

2 Proper Kernelizations

A kernelization is a *proper kernelization* if $g(k) \leq k$ in Def. 1, i.e., we have $k' \leq k$. This additional requirement is backed by the intuition that kernelizations are meant to provide small instances (smaller than the given one), and a blow-up (even if only in the parameter) would counter this intuition. Secondly, as mentioned above, kernelizations are often described purely in terms of reduction rules. As long as $k' \leq k$ and $\text{size}(I') \leq \text{size}(I)$ (and not both inequalities may turn into equalities at the same time) is valid for each kernelization rule, rule-induced kernelizations can be rather easily seen to work in polynomial time; this can

be no longer guaranteed if some rules that constitute the kernelization are not proper. Thirdly, most kernelizations that can be found in the literature are rule-induced and therefore proper. For example, there is only one kernelization among dozens presented in [11] that is not proper, namely, the one for NONBLOCKER SET, also described in [4]. Also observe that rule-induced kernelizations are a neat formalization of the heuristic idea of data reduction that is very successful in practice. Fourthly, (rule-induced) kernelizations have been quite successfully used as a means to speed up search tree algorithms, see [11] for a couple of examples. To this end, kernelizations and branchings are interleaved. If non-proper kernelizations would be used, this would at least complicate the run-time analysis of the obtained algorithm; properness is in fact at least implicitly used to show that applying the reduction rules never worsens the overall running time.

Browsing through the literature, we can often find problem kernels to be defined via proper kernelizations. A quick analysis of the proof sketch of Theorem 1 reveals that this is no loss of generality:

Corollary 1. *A parameterized problem is in \mathcal{FPT} iff it is properly kernelizable.*

It would be interesting to see if also the quality of the kernelization, measured in terms of kernel size, is never worse when insisting on proper kernels. For example, in the case of NONBLOCKER SET, see [4], the best proper kernel we know of is of size $2k$, while the best non-proper (and also the best annotated kernel, see the discussion below) is of size $5/3k$.

3 Parameterized Kernelizations

Conversely, we now discuss the possibility to further generalize the notion of kernelization we had so far. Notice that in terms of a general philosophy, kernelizations can be seen as a sort of “self-reduction” of a problem instance to another instance of the same kind. But is this intuition really reflected in Definition 1? Possibly not, since the proper notion of a reduction in our case would be that of a parameterized reduction. This would mean that in Definition 1, we would weaken the requirement of a kernelization being computable in polynomial time to being computable in \mathcal{FPT} time. This would render Theorem 1 completely trivial, since all work could be done by the reduction.

However, such a notion could make perfect sense from the practical point of view of data reduction: reductions “cheaper” than the final solving methodology should be always used as good preprocessing. In a sense, the customary branching rules that are employed in search tree processing in order to prefer good branches could be likewise seen as intercalated parameterized kernelizations.

When viewing branching rules as kernelizations, we are automatically lead to two further generalizations of the notion of kernelization as discussed so far: Firstly, it is quite natural to consider branchings as sorts of Turing-type reductions instead of many-one reductions. We will not discuss Turing reductions further here, since we did not find any other use of them in parameterized algorithms so far. Secondly, branching might change the nature of the problem by

providing what we will call annotations in the next section. For example, when considering a branching algorithm for DOMINATING SET, putting a vertex x into the dominating set to be constructed causes that the neighborhood $N(x)$ need not be dominated, although $N(x)$ cannot be deleted, since it still might be a good idea to put some $y \in N(x)$ into the dominating set to dominate $N(y)$.

4 Annotated Kernelizations

In the literature, there are cases where kernelization results are claimed that do not fall under the notions discussed so far. For example, in [1], a “kernelization” algorithm for FACE COVER (FC) is presented. However, that reduction takes an instance of FC and produces a kernel instance of ANNOTATED FACE COVER (FCANN). While an instance of FC is just a usual plane graph (plus the parameter bounding the size of the face cover), an instance of FCANN allows annotating (marking) the vertices and faces. Furthermore, an instance of FCANN may contain loops and multiple edges. Observe: any instance of FC can be trivially seen as an instance of FCANN by considering all vertices and faces as unmarked.

We therefore propose the following definition:

Definition 2. *Let \mathcal{P} and \mathcal{P}' be parameterized problems. \mathcal{P}' is called an annotation of \mathcal{P} iff there is a mapping r that maps I onto I' such that for all parameter values k : (I, k) is a YES-instance of \mathcal{P} iff (I', k) is a YES-instance of \mathcal{P}' and r can be implemented by a deterministic finite transducer.*

Recall that a deterministic finite transducer is a deterministic finite automaton with output that may output longer strings than seen in the input. In our example, the transducer may output “unmarked” (besides the mere copying of the input) upon seeing any occurrence of a vertex or face in the input (and this should be viewed as a possible encoding of an annotated instance). Composing two deterministic finite transducers gives a deterministic finite transducer.

Proposition 1. *Let \mathcal{P} , \mathcal{P}' and \mathcal{P}'' be parameterized problems. If \mathcal{P}' is an annotation of \mathcal{P} and if \mathcal{P}'' is an annotation of \mathcal{P}' , then \mathcal{P}'' is an annotation of \mathcal{P} .*

For example, FCANN is an annotation of FC. We could also consider the variant of FCANN that has, in addition, real-number weights $w(f) \geq 1$ associated to each face f . Then, WEIGHTED FCANN is an annotation of FCANN and hence WEIGHTED FCANN is an annotation of FC.

Not all “closely related” problems are annotations. For example, consider the problem TRIANGLE VERTEX DELETION (TVD), first investigated in [14].

Given: A graph $G = (V, E)$

Parameter: a positive integer k

Question: Is there an vertex set $C \subseteq V$ with $|C| \leq k$ whose removal produces a graph without triangles as vertex-induced subgraphs?

We can abuse earlier results [10] of ours to see:

Corollary 2. *TVD can be solved in time $\mathcal{O}((\text{size}(V(G)))^3 + 2.1788^k)$, given G .*

Namely, we can translate every TVD instance (G, k) into a 3-HITTING SET (3-HS) instance by mapping every triangle of G onto a hyperedge of the constructed hypergraph. However, doing this could take cubic time (for example when dealing with a complete graph K_n). This is surely not an annotation (with respect to the usual representation(s) of a graph). However, we could also consider graph representations that explicitly list all triangles in a graph. This shows that the notion of annotation is dependent on the chosen representation of the instance. However, on the very low level, a problem would also fix the coding of the instances (although usually not explicitly given), since only then problems would become formal languages and complexity classes would correspond to formal language classes. In this sense, Def. 2 is sound.

How do we get the running time claimed in Cor. 2? After having translated the TVD instance to the corresponding 3-HS instance, we can run the known kernelization procedure for 3-HS (again in cubic time when measured against the number of vertices, see [16]) and then the search tree algorithm exhibited in [10]. Observe that we cannot claim to have yet a kernel for TRIANGLE VERTEX DELETION, but still we get the “additive parameterized complexity” seen to be typical for \mathcal{FPT} algorithms involving kernelization. We will see in the following section that with additional assumptions it is even possible to inherit a kernelization algorithm for TVD from 3-HS.

We can say that the parameterized problem \mathcal{P} possesses a (proper) annotated kernel if there is an annotation of \mathcal{P}' that possesses a (proper) kernel. Due to the very restricted character of operation we allow for annotation, also the size measures and the running times immediately translate from the known kernelization result for \mathcal{P}' to annotated kernelization results for \mathcal{P} . We will study an example (namely again TVD) in more detail in the following section.

Let us turn to another example, namely nonblocker set (NB): Given a graph $G = (V, E)$ and a positive integer k , is there a *nonblocker set* $N \subseteq V$ with $|N| \geq k$? In [4], it was recently shown how to produce a kernel of size $5/3k + 3$. Without giving the reduction rules here, we like to mention that in fact we provided a kernel of size $5/3k$ for the annotation where, in addition, a vertex d is specified and we require that $d \notin N$.

In that particular example, we could produce a reduction rule that gets rid of the annotation (called catalyzation in [4]) at the expense of introducing three more vertices to the kernel. So, this gives us an example where a kernel for an annotated version could be used to produce a kernel for the original version of the problem. It would be interesting to see if there are more examples along this venue or if the notion of an annotated kernel should be standing as a notion on its own right.

5 Induced Kernels

The main motivation for this section is to provide a framework that allows to transfer kernelization results (in the classical sense) from one problem to another. A further motivation for studying linear-size induced kernels would be

the possibility to use exponential space in order to improve on the running times. Let us continue our discussion of the relationship between TVD and 3-HS as a running example. Let us first repeat the reduction rules for 3-HS from [16] in a slightly modified (and corrected) form:

1. (hyper)edge domination: A hyperedge e is *dominated* by another hyperedge f if $f \subset e$. In that case, delete e .
2. tiny edges: Delete all hyperedges of degree one and place the corresponding vertices into the hitting set (reducing the parameter accordingly).
3. vertex domination: A vertex x is *dominated* by a vertex y if, whenever x belongs to some hyperedge e , then y also belongs to e . Then, we can simply delete x from the vertex set and from all edges it belongs to.
4. too many edges containing a vertex pair: If $\{x, y\}$ is a pair of vertices such that the number of edges that contain both x and y exceeds k , then add an edge $\{x, y\}$.
5. too many edges containing a vertex: If x is a vertex such that the number of edges that contain x exceeds k^2 , then add an edge $\{x\}$.

Notice that the last two rules only seemingly increase the size of the instance, since immediately one of the first two rules would be triggered. Hence, we could reformulate them as follows:

- 4' too many edges containing a vertex pair: If $\{x, y\}$ is a pair of vertices such that the number of edges that contain both x and y exceeds k , then delete all edges containing both x and y and replace them by an edge $\{x, y\}$.
- 5' too many edges containing a vertex: If x is a vertex such that the number of edges that contain x exceeds k^2 , then delete all edges containing x and reduce the parameter by one.

Now, if none of the above rules applies to a 3-HS instance, the following cutting rule applies:

- If the instance has more than k^3 hyperedges, then NO.

The cutting rule implies that there is a $\mathcal{O}(k^3)$ rule-induced kernel for 3-HS. Annotated kernels. One immediate problem with the idea of translating these rules into rules for TVD is that only hyperedges with exactly three vertices have an interpretation as “triangles” in the TVD “world.” This is still true when observing that rules 4' and 5' together with the cutting rule are sufficient to provide the known kernelization result for 3-HS, since rule 4' might introduce hyperedges of size two even if the original instance only contains hyperedges of size three (this would correspond to a 3-HS instance obtained from a given TVD instance). One way out of this sort of dilemma might be a specific form of annotation of graphs: in this sense, an ANNOTATED TVD instance would consist of specifying a graph $G = (V, E)$, a parameter k and another graph $G' = (V, E')$ specified by E' , and the question would be to find a vertex set C , $|C| \leq k$ that is a vertex cover of G' and whose removal destroys all triangles of G . What would be the kernelization rules for ANNOTATED TVD?

1. Delete vertices of degree zero in G that are not contained in edges from E' .
2. If x is a vertex of degree one in G , then delete the incident edge from E .
3. If $\{x, y\}$ is a pair of vertices such that the number of triangles in G that contain both x and y exceeds k , then move the edge $\{x, y\}$ from E into E' . (Hence, we “remove” the triangles containing x and y from E .)
4. If x is a vertex such that the number of triangles that contain x exceeds k^2 , then delete all edges containing x from E and from E' and reduce the parameter by one.

Now, if none of the above rules applies to the ANNOTATED TVD instance, the following cutting rule applies:

- If the reduced instance has more than k^3 triangles, then NO.

Recently, a very similar kernelization for TRIANGLE VERTEX DELETION was proposed in [6], including suggesting a proper de-annotation rule. However, the kernel obtained that way is not a subgraph of the original graph, in contrast to our construction given below.

Another case where an annotated kernel could be smaller than a non-annotated kernel is EDGE DOMINATING SET, as presented in another paper of second author in these proceedings.

Induced kernels. A subtle drawback of the given kernelization for 3-HS is the fact that a reduced instance is obtained by changing some edges without deleting their vertices. In other words, the kernel is a subgraph that is not vertex-induced. This complication may seem harmless in general, but could be serious when the HS instance is used to model problems in such a way that certain forbidden relations between the objects (vertices) translate to edges in the input to HS.

We offer a new kernelization algorithm to remedy this situation at the cost of more involved reduction rules. We will also use the tiny edge rule and isolated vertex rule to obtain the kernel. However, we cannot use the hyperedge domination rule, since this rule destroys inducedness. Whenever we face an instance of 3-HITTING SET, we define: Let $F = \{x \in V \mid x \text{ shares more than } k \text{ edges with some } y \text{ in } V\}$ and let S be the complement of F in V . Elements of F and S will be referred to in the sequel as *fat* and *slim* vertices, respectively. The set of (hyper-)edges E can be partitioned as follows:

$$E_{j*s} = \{e \in E \mid |e| = 3 \wedge |e \cap S| = j\} \quad \text{for } j = 0, 1, 2, 3; \text{ and}$$

$$E_p = \{e \in E \mid |e| = 2\}.$$

Observe that edges that only contain one vertex can be dealt with by using the tiny edge rule, so that in fact we can assume that $E = \bigcup_{j=0}^3 E_{j*s} \cup E_p$.

Let us define the *co-occurrence* of a pair $\{x, y\}$ of vertices to be the number of edges that “contain” the two vertices simultaneously. Denote by $co(x, y)$ the co-occurrence of $\{x, y\}$. We say that x, y co-occur iff $co(x, y) > 0$. In the rules listed below, whenever we say that we put x into the hitting set, then this means that we reduce the parameter by one and delete x and all edges containing x from the instance. Denote by H the target hitting set. In order to justify the

actions taken by the following reduction rules, we shall assume the input is a YES-instance (so, $|H| \leq k$).

1. If x is a vertex that occurs more than k times in edges from E_p , then put x into the hitting set.

The soundness of this first rule is obvious. If x is not in H , then its (more than k) neighbors in E_p are all needed to cover the edges of E_p .

2. If x is a slim vertex of degree larger than k^2 , then put x into the hitting set.

If x does not belong to H , then the degree of x must be bounded above by k^2 . Otherwise, x appears more than k times with an element of H (which violates the definition of S). Therefore this second rule is sound.

3. If x is a fat vertex that appears more than k times with more than k different other fat vertices, then put x into the hitting set.

To see this, note that if $co(x, y) > k$, then at least one element of $\{x, y\}$ must be in H . It follows that if x is excluded from H , then more than k elements (co-occurring more than k times with x) would be needed in H , a contradiction to the assumption that the input is a YES-instance.

4. If x is a fat vertex that belongs to more than k^2 edges of E_{2*s} , then put x into the hitting set.

If x were not in H , then its edges in E_{2*s} must be hit by slim vertices only. Since it appears in more than k^2 such edges, we conclude that a slim neighbor of s (in E_{2*s}) must have more than k common edges with x , contradicting the definition of slim vertices.

Consider the simple graph G_F , constructed as follows: (i) $V(G_F) = F$, and (ii) $E(G_F) = \{(u, v) : co(u, v) > k\}$. Then G_F must have a vertex cover of size k or less. It follows that, after applying rule 3 above, F contains at most $k^2 + k$ vertices. A similar argument shows that the total number of vertices that appear in E_p is bounded above by $k^2 + k$. Moreover, every vertex appears in at most k^2 edges of $E_{2*s} \cup E_{3*s}$. Therefore, unless we have a NO instance, the number of edges in $E_{2*s} \cup E_{3*s}$ is bounded above by k^3 (H has at most k vertices, each appearing in at most k^2 edges). So, it remains to find an upper bound on the number of edges (or slim vertices) appearing in E_{1*s} . To do this, we partition E_{1*s} further. Let $E_{1*s,<} = \{\{x, y, z\} \in E_{1*s} \mid x, y \in F \text{ and } co(x, y) \leq k \text{ in } E\}$. Moreover, let $E_{1*s,>} = E_{1*s} \setminus E_{1*s,<}$.

- If x is a vertex that occurs more than k^2 times in edges of $E_{1*s,<}$, then put x into the hitting set.

Let x be any vertex that belongs to more than k^2 edges in $E_{1*s,<}$. If $x \notin H$, then x would share more than k edges (in $E_{1*s,<}$) with another vertex, violating the definitions of $E_{1*s,<}$ or S (depending on whether x is fat vertex appearing more than k times with another fat vertex in $E_{1*s,<}$ or x is slim one appearing more than k times with another vertex). Therefore, $E_{1*s,<}$ has at most k^3 edges.

- If x is a slim vertex that occurs in edges of $E_{1*s,>}$ and does not appear elsewhere in E , then delete x .

Any such slim vertex would be dominated. Every edge that contains such a vertex is guaranteed to be covered by some fat vertex.

Finally, we have showed that, after applying our reduction rules, the number of edges containing slim vertices is bounded above by $2k^3$ (k^3 in $E_{1*s,<}$ and k^3 in $E_{2*s} \cup E_{3*s}$). Therefore, the number of slim vertices is in $O(k^3)$ (due to the vertex domination rule, the upper bound is $3k^3$). Since the number of fat vertices is quadratic in k , we can conclude:

Theorem 2. 3-HITTING SET admits a vertex-induced problem kernel of size $O(k^3)$, measured both in the number of vertices and in the number of edges.

It is tedious but possible to generalize our approach to obtain induced kernels for d -HS for any $d > 2$.

Corollary 3. TVD admits a problem kernel of size $O(k^3)$.

Proof. (Sketch) Since all rules presented above only delete vertices or conclude that we face a NO-instance, the rules can be immediately interpreted as TVD reduction rules.

We have used a similar translation to provide the first ever published small-size kernel for ONE-LAYER PLANARIZATION, a problem arising in the area of graph drawing, see [13].

The reader might have noticed that we have not yet given a proper general definition of what we might mean by “induced kernels.” This notion is quite clear for (hyper)graphs (in particular for vertex set minimization problems, vertex-induced would be the appropriate formalization), but less clear otherwise.

To provide our general notion of induced kernels, we consider only \mathcal{FPT} problems \mathcal{P} that are *naturally-parameterized*, meaning the following:

- \mathcal{P} is parameterized only by its target solution size k , and
- every instance I of \mathcal{P} contains a set of distinguished elements that qualify for membership in the solution set. Such elements form what we call the *candidate set* and what we denote $\kappa(I)$.
- Finally, we require that $k \leq |\kappa(I)| \leq \text{size}(I)$.

In the case of HITTING SET, the vertex set would be the candidate set. Let \mathcal{P} be a naturally-parameterized \mathcal{FPT} problem and (I, k) be an instance of \mathcal{P} . A kernelization K for \mathcal{P} is said to *produce induced kernels* if, on (I, k) , K outputs:

1. the reduced instance (I', k') with $k' \leq k$;
2. a *partial solution* set $S \subseteq \kappa(I)$ with $\kappa(I') \cap S = \emptyset$ of size $k - k'$ consisting of elements of the candidate set that are selected into the solution;
3. a *trash* set $T \subseteq \kappa(I)$ with $(\kappa(I') \cup S) \cap T = \emptyset$.

In the 3-HS example, the problem is naturally-parameterized and the candidate set is the set of vertices. Also note that our vertex-induced kernel of 3-HS does not entail any direct deletion of edges. In other words, constraints that relate a set of candidates will be deleted (automatically) only when the candidates are removed. In the case of VC, the Nemhauser-Trotter reduction explicitly provides the sets S and T , see [11]. This generalized concept of induced kernels yields an important relationship between hardness of approximation and *hardness of kernelization*.

Theorem 3. *Let \mathcal{P} be a minimization problem. If the naturally-parameterized version of \mathcal{P} has a kernelization algorithm that produces, for any instance (I, k) , an induced kernel (I', k') such that $\text{size}(I') \leq \alpha \cdot k'$ for some constant $\alpha > 1$, then \mathcal{P} has an approximation algorithm with ratio α .*

Proof. Let I be an instance of \mathcal{P} for which we seek an optimum solution. Let K be a kernelization algorithm that satisfies the statement of this theorem. The corresponding approximation algorithm $A_K(k)$ works as follows: (1) produce $I'(k) = K(I, k)$; let $S(k)$ be the partial solution and $T(k)$ the trash set obtained as a by-product of applying K ; and (2) output $S(k) \cup \kappa(I'(k))$. Notice that the output of $A_K(k)$ is a feasible solution (by the definition of an induced kernel, since elements of T are excluded from some solution that contains S).

Let $S_{opt} \subseteq \kappa(I)$ be an optimum solution of the instance I . Then,

$$\frac{|S(k) \cup \kappa(I'(k))|}{|S_{opt}|} \leq \frac{k - k' + \alpha \cdot k'}{|S_{opt}|} \leq \frac{\alpha \cdot k}{|S_{opt}|}$$

This is not yet quite our result, but we can iterate $A_K(k)$ for all $k=0, \dots, |\kappa(I)|$. The parameter k that yields the smallest value of $|S(k) \cup \kappa(I'(k))|$ surely satisfies $|S(k) \cup \kappa(I'(k))| \leq |S(k_{opt}) \cup \kappa(I'(k_{opt}))|$, where $k_{opt} = |S_{opt}|$. Hence, an algorithm A_K that finally outputs the smallest solution found by any of the $A_K(k)$ is an α -approximation. Notice that the overall algorithm runs in polynomial time, since K does so and $|\kappa(I)| \leq \text{size}(I)$.

It was shown in [5] that VERTEX COVER is hard to approximate with a ratio bound of ≈ 1.36 . Combining this result with Theorem 3 yields the following.

Corollary 4. *Unless $\mathcal{P} = \mathcal{NP}$, we can claim: for any $\epsilon > 0$, VERTEX COVER does not have a $(1.36 - \epsilon)k$ induced kernel.*

6 Kernelization Schemes

In the design of approximation algorithms, it has been quite popular to design *algorithm schemes*, i.e., families of algorithms that, in the case of approximation, provide better and better approximation guarantees, at the expense of higher and higher running times. In the case of approximation algorithms, such schemes were called *approximation schemes*. Is it possible to translate this idea into the realm of parameterized algorithmics?

To this end, let us turn to LINEAR ARRANGEMENT (LA), see [12,15,18] for a treatment from a parameterized perspective. Given a graph $G = (V, E)$ and a positive integer k , is there a one-to-one mapping $\sigma : V \rightarrow \{1, \dots, |V|\}$ such that $\sum_{\{u,v\} \in E} |\sigma(u) - \sigma(v)| \leq k$? It is easily seen that LA is fixed parameter tractable, based on the following observation cast in the form of reduction rules that immediately give a problem kernel:

Rule 1. *Let (G, k) be an instance of LA. (a) If v is an isolated vertex, then reduce to $(G - v, k)$. (b) If e is an isolated edge, then reduce to $(G - e, k)$.*

Namely, the endpoints of an isolated edge can be arbitrarily arranged.

Rule 2. *If (G, k) is an instance of LA and if $G = (V, E)$ is reduced with respect to Rule 1, then return NO if $|V| > 1.5k$.*

Proposition 2. *A reduced instance of LA has at most $1.5k$ vertices.*

Our reduction rules can be generalized: optimal arrangements for all graphs up to $q-1$ vertices may be precomputed (taking exponential time, measured against q) and stored in a huge table; this shows that each component then has at least q vertices. With an appropriate reduction rule, we may deduce:

Proposition 3. *For fixed q , a reduced instance of LA has at most $\frac{q}{q-1}k$ vertices.*

This certainly has the flavor of an algorithmic scheme: at the expense of larger and larger running times, we may get smaller and smaller kernels (let q grow). Notice however that it is not at all clear whether the bound k is sharp in some sense. This (still) contrasts the analogous idea of approximation schemes. We have encountered a similar scheme for other problems, as well, e.g., for POSITIVE WEIGHTED COMPLETION OF AN ORDERING, see [9]. Finally, in [2] generalizations of reduction rules for PLANAR DOMINATING SET have been presented; unfortunately, it is unknown if their so-called data reduction scheme is really providing smaller kernels. It would be interesting to see examples of problems that admit kernelization schemes that actually “converge” to a provable lower bound.

7 Conclusions

We have tried to raise some questions concerning one of the very basic notions of parameterized complexity theory: namely, kernels. This is rather a conceptual paper that classifies kernelization strategies found in practice than a paper presenting many mathematical results. However, we hope to stir discussions of the notions suggested in this paper. We also raised many open questions regarding the notions. The most important conceptual ones are:

- Is “properness” a real restriction for kernel sizes?
- Does “annotation” really give more flexibility?
- What are good “master problems” to work on induced kernels?
- Can we get lower bound results on kernel sizes stronger than what was presented in [3] or in this paper, or possibly with fewer assumptions?
- Are there good examples of kernelization schemes (with sharp lower bounds)?

More generally speaking, the relationships between parameterized complexity and approximability need further research.

Acknowledgment. We are grateful for discussions with Peter Shaw.

References

1. F. Abu-Khazam and M. Langston. A direct algorithm for the parameterized face cover problem. In *International Workshop on Parameterized and Exact Computation IWPEC 2004*, volume 3162 of *LNCS*, pages 213–222. Springer, 2004.
2. J. Alber, B. Dorn, and R. Niedermeier. A general data reduction scheme for domination in graphs. In *Software Seminar SOFSEM*, volume 3831 of *LNCS*, pages 137–147. Springer, 2006.
3. J. Chen, H. Fernau, I. A. Kanj, and Ge Xia. Parametric duality and kernelization: Lower bounds and upper bounds on kernel size. In *Symposium on Theoretical Aspects of Computer Science STACS 2005*, volume 3404 of *LNCS*, pages 269–280. Springer, 2005.
4. F. Dehne, M. Fellows, H. Fernau, E. Prieto, and F. Rosamond. NONBLOCKER: parameterized algorithmics for MINIMUM DOMINATING SET. In *Software Seminar SOFSEM*, volume 3831 of *LNCS*, pages 237–245. Springer, 2006.
5. I. Dinur and S. Safra. On the hardness of approximating minimum vertex cover. *Annals of Mathematics*, 162:439–485, 2005.
6. M. Dom, J. Guo, F. Hüffner, R. Niedermeier, and A. Truß. Fixed-parameter tractability results for feedback set problems in tournaments. In *Conference on Algorithms and Complexity CIAC*, volume 3998 of *LNCS*, pages 320–331. Springer, 2006.
7. R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Springer, 1999.
8. R. G. Downey, M. R. Fellows, and U. Stege. Parameterized complexity: A framework for systematically confronting computational intractability. In *Contemporary Trends in Discrete Mathematics: From DIMACS and DIMATIA to the Future*, pages 49–99. 1999.
9. V. Dujmović, H. Fernau, and M. Kaufmann. Fixed parameter algorithms for one-sided crossing minimization revisited. In *Graph Drawing, 11th International Symposium GD 2003*, volume 2912 of *LNCS*, pages 332–344. Springer, 2004.
10. H. Fernau. A top-down approach to search-trees: Improved algorithmics for 3-Hitting Set. Technical Report TR04-073, Electronic Colloquium on Computational Complexity ECCC, 2004.
11. H. Fernau. *Parameterized Algorithmics: A Graph-Theoretic Approach*. Habilitationsschrift, Universität Tübingen, Germany, 2005.
12. H. Fernau. Parameterized algorithmics for linear arrangement problems. In *CTW 2005: Workshop on Graphs and Combinatorial Optimization*, pages 27–31. University of Cologne, Germany, 2005. Long version submitted.
13. H. Fernau. Two-layer planarization: improving on parameterized algorithmics. *Journal of Graph Algorithms and Applications*, 9:205–238, 2005.
14. J. Gramm, J. Guo, F. Hüffner, and R. Niedermeier. Automated generation of search tree algorithms for hard graph modification problems. *Algorithmica*, 39:321–347, 2004.
15. G. Gutin, A. Rafiey, S. Szeider, and A. Yeo. The linear arrangement problem parameterized above guaranteed value. In *Conference on Algorithms and Complexity CIAC*, volume 3998 of *LNCS*, pages 356–367. Springer, 2006.
16. R. Niedermeier and P. Rossmanith. An efficient fixed-parameter algorithm for 3-Hitting Set. *Journal of Discrete Algorithms*, 1:89–102, 2003.
17. R. Niedermeier and P. Rossmanith. On efficient fixed parameter algorithms for weighted vertex cover. *Journal of Algorithms*, 47:63–77, 2003.
18. M. Serna and D. M. Thilikos. Parameterized complexity for graph layout problems. *EATCS Bulletin*, 86:41–65, 2005.