Analysis of Data Reduction: Transformations give evidence for non-existence of polynomial kernels

Hans L. Bodlaender

Stéphan Thomassé

Anders Yeo

Technical Report UU-CS-2008-030
September 2008

Department of Information and Computing Sciences
Utrecht University, Utrecht, The Netherlands
www.cs.uu.nl

ISSN: 0924-3275

Department of Information and Computing Sciences
Utrecht University
P.O. Box 80.089
3508 TB Utrecht
The Netherlands

Analysis of Data Reduction: Transformations give evidence for non-existence of polynomial kernels

Hans L. Bodlaender[*] Stéphan Thomassé[†] Anders Yeo[‡]

Abstract

In this paper, we introduce a new technique to give evidence for combinatorial problems that they cannot be preprocessed in polynomial time such that resulting instances always have a size bounded by a polynomial in a specified parameter (or, in short: do not have a polynomial kernel); these results are assuming the validity of certain complexity theoretic assumptions. We build upon a framework by Bodlaender et al. [6], and add a notion of transformation to this framework. Using these transformations, we show that Disjoint Cycles, and Disjoint Paths do not have polynomial kernels, unless the or-distillation conjecture does not hold, which would imply by a result of Fortnow and Santhanam [12] that $NP \subseteq coNP/poly$, and we show that Hamiltonian Circuit parameterized by treewidth does not have a polynomial kernel, unless the and-distillation conjecture does not hold. We also show that the problem to determine if there are k edge disjoint cycles has a polynomial kernel.

1 Introduction

In many practical settings, exact solutions to NP-hard problems are needed. A common approach in such cases is to start with a preprocessing or data reduction algorithm: before employing a slow exact algorithm (e.g., ILP, branch and bound), we try to transform the input to an equivalent, smaller input.

Currently, the theory of fixed parameter complexity gives us tools to make a theoretical analysis of such data reduction or preprocessing algorithms. A *kernelization* algorithm is an algorithm, that uses polynomial time, and transforms an input for a specific problem to an equivalent input whose size is bounded by some function of a parameter. The resulting

[*]Institute of Information and Computing Sciences, Utrecht University, the Netherlands. hansb@cs.uu.nl
[†]LIRMM-Université Montpellier II, 161 Rue Ada, 34392 Montpellier Cesex, France. thomasse@lirmm.fr
[‡]Departmente of Computer Science, Royal Holloway University of London, Egham, Surrey TW20 OEX, United Kingdom. anders@cs.rhul.ac.uk

instance is also called a *kernel.* Questions of both theoretical and practical interests are for a specific problem: does it have a kernel, and if so, how large can this kernel be? An excellent overview of much recent work on kernelization was made by Guo and Niedermeier [14].

For the question, whether a specific (parameterized) problem has a kernel, the fixed parameter tractability theory introduced by Downey and Fellows gives good tools to answer these. We say a problem is fixed parameter tractable (in FPT), if it has an algorithm that runs in time $O(n^c f(k))$, with n the input size, k the parameter, c a constant, and f any function. Now, it can easily be seen that a decidable problem is in FPT, if and only if it has a kernel.[1]

Recently, Bodlaender and al. [6] gave a framework to give evidence that problems (in FPT) do not have a kernel of polynomial size. The framework is based upon the notion of *composability.* There are actually two forms: and-composability, and or-composability. We have a parameterized problem, whose variant as a decision problem is NP-complete, and it is and-composable, then it does not have a kernel whose size is bounded by a polynomial, unless the and-distillation conjecture does not hold. Similarly for or-composability, and the or-distillation conjecture, but in this case, one can use a result by Fortnow and Santhaman, and strengthen the conjecture to $NP \not\subseteq coNP/poly$ [12].

In this paper, we extend the framework by introducing a notion of *transformation.* While the main idea parallels classic notions of transformation, we think that our contribution is a new important tool for the theory of data reduction/kernelization and fixed parameter tractability. We use our framework to show for the following problems that they do not have a kernel of polynomial size unless $NP \not\subseteq coNP/poly$: Disjoint Cycles, Disjoint Paths, Hamiltonian Circuit parameterized by treewidth. The latter problem is an example of more problems where similar techniques work.

Concerning the size of a kernel of a parameterized problem, we can summarize the situation in Table 1. Assuming that the problem is decidable, the second and third column give the main available positive or negative, respectively, evidence that the problem has a kernel of the size given in the first column. E.g., $W[1]$-hardness indicates that a problem is not in FPT; a problem is in FPT, if and only if it has a kernel of any size (i.e., bounded by a function of k.) The use of transformations is a contribution made by this paper.

This paper is organized as follows. In Section 2, we give some known results and introduce some new notations. In Section 3, we introduce the notion of polynomial parameter transformation, and show how it can be used to obtain results on the existence of polynomial kernels. In Section 4, we apply the technique to some concrete problems. In particular, we show that Disjoint Cycles, Disjoint Paths, and Hamiltonian circuit with treewidth as parameter have no polynomial kernel unless $NP \subseteq coNP/poly$. We also give a short proof that the variant of Disjoint Cycles where cycles must be *edge* disjoint does have a polynomial kernel. Section 5 concludes the paper with some final

[1]We sketch this folklore result: If the problem has a kernel, then decide the kernel. If a problem has an $O(n^c f(k))$ algorithm, then if $n \leq f(k)$, do nothing, and otherwise the algorithm solves the problem in $O(n^{c+1})$ time, and transform to an $O(1)$-size yes- or no-instance.

size	**positive evidence**	**negative evidence**	conjecture
$O(1)$	P-time algorithm	NP-hardness	$P \neq NP$
polynomial	poly-kernel algorithm	composability & NP-c **transformations**	$NP \not\subseteq coNP/poly$, ADC
any kernel	$\in$ FPT	$W[1]$-hardness	$FPT \neq W[1]$ ETH

Table 1: Size of kernels and evidence. ETH = Exponential Time Hypothesis. ADC = And-distillation Conjecture. NP-c = NP-completeness

remarks.

2 Notions

In this section, we give several results, mostly from [6], and introduce some new notation. We also give some basic notions from fixed parameter tractability, as introduced by Downey and Fellows, see e.g., [11].

Definition 1 *A parameterized problem is a subset of $L^* \times \mathbf{N}$ for some finite alphabet L: the second part of the input is called the* parameter*.*

We assume here that the second parameter is an integer. It is not hard to modify the techniques such that it works with other types of parameters, e.g., pairs of integers.

Definition 2 *A parameterized problem $Q \subseteq L^* \times \mathbf{N}$ is said to belong to the class FPT (to be* fixed parameter tractable*, if there is an algorithm A, a polynomial p, and a function $f : \mathbf{N} \to \mathbf{N}$, such that A determines for a given pair $(x, k) \in L^* \times \mathbf{N}$ whether $(x, k) \in Q$ in time at most $p(|x|) \cdot f(k)$. ($|x|$ denotes the length of input x.)*

Definition 3 *A* kernelization algorithm *for a parameterized problem $Q \subseteq L^* \times \mathbf{N}$ computes a function $K : L^* \times \mathbf{N} \to L^* \times \mathbf{N}$, such that*

- *For all $(x, k) \in L^* \times \mathbf{N}$, the algorithm takes time polynomial in $|x| + k$.*
- *For all $(x, k) \in L^* \times \mathbf{N}$: $(x, k) \in Q \Leftrightarrow A(x, k) \in Q$.*
- *There is a function $g : \mathbf{N} \to \mathbf{N}$, such that for all $(x, k) \in L^* \times \mathbf{N}$: $|A(x, k)| + k \leq g(k)$.*

We say that Q has a *kernel of size f*. If f is bounded by a polynomial in k, we say that Q has a *polynomial kernel*.

Sometimes, in the literature one requires that the kernelization algorithm does not increase the parameter, i.e., when we write $A(x, k) = (x', k')$, then $k' \leq k$. This assumption is not necessary for obtaining the results and deleting it slightly strengthens our results.

Bodlaender et al. [6] give the following two conjectures.

Conjecture 4 (And-distillation conjecture [6]) *Let R be an NP-complete problem. There is no algorithm D, that gets as input a series of m instances of R, and outputs one instance of R, such that*

- *If D has as input m instances, each of size at most n, then D uses time polynomial in m and n, and its output is bounded by a function that is polynomial in n.*
- *If D has as input instances $x_1, \ldots, x_m$, then*

$$D(x_1, \ldots, x_m) \in R \Leftrightarrow \forall_{1 \leq i \leq m} x_i \in R.$$

Conjecture 5 (Or-distillation conjecture [6]) *Let R be an NP-complete problem. There is no algorithm D, that gets as input a series of m instances of R, and outputs one instance of R, such that*

- *If D has as input m instances, each of size at most n, then D uses time polynomial in m and n, and its output is bounded by a function that is polynomial in n.*
- *If D has as input instances $x_1, \ldots, x_m$, then*

$$D(x_1, \ldots, x_m) \in R \Leftrightarrow \exists_{1 \leq i \leq m} x_i \in R.$$

Theorem 6 (Fortnow and Santhaman [12]) *If the or-distillation conjecture does not hold, then $NP \subseteq coNP/poly$.*

There is no equivalent to Theorem 6 known for and-distillation. This is an important open problem in this area.

The main tool to give evidence for the non-existence of polynomial kernels for specific parameterized problems from [6] is the notion of compositionality. Compositionality allows us to build one instance from a collection of instances. There are two different notions: and-compositionality and or-compositionality. In the first case, the new instance is a yes-instance, if and only if each instance in the collection is a yes-instance; in the second case, this happens, if and only if at least one instance in the collection is a yes-instance.

Definition 7 *An* and-composition *algorithm for a parameterized problem $Q \subseteq L^* \times \mathbf{N}$ is an algorithm, that gets as input a sequence $((x_1, k), \ldots, (x_r, k))$, with each $(x_i, k_i) \in L^* \times \mathbf{N}$, and outputs a pair (x', k'), such that*

- *the algorithm uses time polynomial in $\sum_{1 \leq i \leq r} |x_i| + k$;*
- *k' is bounded by a polynomial in k*
- *$(x', k') \in Q$, if and only if for all i, $1 \leq i \leq r$, $(x_i, k) \in Q$.*

The definition for *or-composition* is identical, except that the last condition becomes:

- $(x', k') \in Q$, if and only if there exists an i, $1 \leq i \leq r$, $(x_i, k) \in Q$.

Many problems have natural composition algorithms. For many graph problems, the only operation needed is the disjoint union of connected components. Consider for instance the Longest Cycle problem: does G have a cycle of length at least k? As a graph has a cycle of length at least k, if and only if at least one of its connected components has such a cycle, the problem is trivially or-compositional.

We need one further notion: for a parameterized problem, we have the *derived classic problem.* Formally, if $R \subseteq L^* \times \mathbf{N}$ is a parameterized problem, we take a symbol $\mathbf{1} \notin L$, and take as derived classic problem the set $\{x\mathbf{1}^k \mid (x, k) \in R\}$. Here, we associate in a natural way a classic one-argument input problem with a parameterized problem; note that we assume that the parameter is given in unary. For instance, the Disjoint Cycles problem as parameterized problem belongs to FPT, and its derived classic problem is NP-complete. In several cases, we use the same name for the derived classic problem as for the parameterized version.

We now give here some results from [6] and other papers, and introduce some notation (the classes NPK^0_{or} and NPK^0_{and}) that will be helpful for further presentation of the results.

Definition 8 *The class NPK^0_{and} is the class of parameterized problems, that are and-compositional and whose derived classical problem is NP-complete.*

Definition 9 *The class NPK^0_{or} is the class of parameterized problems, that are or-compositional and whose derived classical problem is NP-complete.*

Theorem 10 (Bodlaender et al. [6])
(i) If a problem in the class NPK^0_{and} has a polynomial kernel, then the and-distillation conjecture does not hold.
(ii) If a problem in the class NPK^0_{or} has a polynomial kernel, then the or-distillation conjecture does not hold.

As a corollary of Theorems 10 and Theorem 6, we have

Corollary 11 (Bodlaender et al.[6], Fortnow and Santhaman [12]) *If a problem in the class NPK^0_{or} has a polynomial kernel, then $NP \subseteq coNP/poly$.*

In turn, $NP \subseteq coNP/poly$ would imply a collapse of the polynomial time hierarchy to the third level. Currently, there is no equivalent result to Theorem 6 known for the and-distillation conjecture.

3 Polynomial time and parameter transformations

We now introduce a notion of transformation, that allows us to prove results for problems that do not obviously have compositionality.

Definition 12 *Let P and Q be parameterized problems. We say that P is polynomial time and parameter reducible to Q, written $P \leq_{Ptp} Q$, if there exists a polynomial time computable function $f : \{0,1\}^* \times \mathbf{N} \rightarrow \{0,1\}^* \times \mathbf{N}$, and a polynomial $p : \mathbf{N} \rightarrow \mathbf{N}$, and for all $x \in \{0,1\}^*$ and $k \in \mathbf{N}$, if $f((x,k)) = (x',k')$, then the following hold:*

- *$(x,k) \in P$, if and only if $(x',k') \in Q$, and*
- *$k' \leq p(k)$.*

We call f a polynomial time and parameter transformation from P to Q.

If P and Q are parameterized problems, and P^c and Q^c are the derived classical problems, then f can also be used as a polynomial time transformation (in the usual sense of the theory of NP-completeness) from P^c to Q^c. As an additional condition to polynomial time transformations, we have that the size of the parameter can grow at most polynomially. Note that the fixed parameter reductions by Downey and Fellows (see e.g., [9, 10, 11]) are similar, but allow non-polynomial growth of the parameter, and are used for a different purpose: to show hardness for $W[1]$ or a related class, thus, these transformations are usually applied to problems that do not have any parameter at all.

Theorem 13 *Let P and Q be parameterized problems, and suppose that P^c and Q^c are the derived classical problems. Suppose that Q^c is NP-complete, and $P^c \in NP$. Suppose that f is a polynomial time and parameter transformation from P to Q. Then, if Q has a polynomial kernel, then P has a polynomial kernel.*

Proof: Suppose that Q has a polynomial kernel. Now, consider the following algorithm, that gets as input a pair $(x,k) \in \{0,1\}^* \times \mathbf{N}$, which is an input for P. First, we compute $f((x,k))$, say $f((x,k)) = (x',k')$. Then, we apply the polynomial kernelization algorithm for Q to (x',k'); suppose this gives (x'',k''). As P^c is NP-complete, there is a polynomial time transformation from Q to P, say f'. Suppose $f'((x'',k'')) = (y,\ell)$.

We claim that the algorithm that transforms (x,k) to (y,ℓ) is a polynomial kernel for P. Note that k' is polynomially bounded in k, as f is a polynomial time and parameter transformation. Now, k'' and the size of x'' are polynomially bounded in k' as these are obtained by a polynomial kernelization algorithm, and thus k'' is polynomially bounded in k. As f' is a polynomial time transformation, and k'' is seen to be in unary notation as input for Q^c, we have that the size of y and ℓ are polynomially bounded in the size of x'' and k'' and hence in k.

It is now easy to see that the algorithm uses polynomial time. Finally, we have that $(x,k) \in P$, if and only if $(x',k') \in Q$, if and only if $(x'',k'') \in Q$, if and only if $(y,\ell) \in P$. $\Box$

Note that we could instead require that P^c is NP-complete, and Q^c is in NP, as the NP-completeness of Q^c would follow from the fact that f is also a "classic" polynomial time transformation.

Corollary 14
(i) Let P and Q be parameterized problems, and suppose that P^c and Q^c are the derived classical problems. Suppose that P and Q are NP-complete, that Q is and-compositional, and that $P \leq_{Ptp} Q$. If P has a polynomial kernel, then the and-compositionality conjecture does not hold.
(ii) Let P and Q be parameterized problems, and suppose that P^c and Q^c are the derived classical problems. Suppose that P and Q are NP-complete, that Q is or-compositional, and that $P \leq_{Ptp} Q$. If P has a polynomial kernel, then the and-compositionality conjecture does not hold, and thus $coNP \subseteq NP/poly$, and thus the polynomial time hierarchy collapses to the third level.

We now define the classes NPK_{or} and NPK_{and} as the closures of NPK^0_{or} and NPK^0_{and} under polynomial time and parameter transformations. Membership in these classes gives a strong indication that there is no polynomial kernel for the problem. We can reformulate the discussion above as (using results from [6] and [12] and our own observations):

Corollary 15
(i) If parameterized problem $P \in NPK_{or}$, then P has no polynomial kernel, unless $coNP \subseteq NP/poly$.
(ii) If parameterized problem $P \in NPK_{and}$, then P has no polynomial kernel, unless the and-distillation conjecture does not hold.

The polynomial time and parameter transformations thus give us a nice method to show unlikeliness of the existence of polynomial kernels.

4 Results for concrete problems

4.1 Disjoint cycles and paths

Consider the following two parameterized problems.

DISJOINT CYCLES
Input: Undirected graph $G = (V, E)$
Parameter: Integer k
Question: Does G contain at least k vertex-disjoint cycles?

The problem is strongly related to the FEEDBACK VERTEX SET problem, which has a kernel of size $O(k^2)$ [20] by Thomassé, who improved upon a kernel of size $O(k^3)$ [5]. Another related problem is whether a given graph contains at least k cycles which are *edge* disjoint. This problem, called DISJOINT CYCLE PACKING has a polynomial kernel. We give a short proof of this new result here. We conjecture that a more refined construction can give a smaller kernel. Techniques resemble techniques from [5, 20]. A related result is a kernel for the version where each cycle must be of length exactly three (EDGE DISJOINT TRIANGLE PACKING) by Mathieson et al. [17].

Theorem 16 Disjoint Cycle Packing *has a kernel with* $O(k^2 \log^2 k)$ *vertices.*

Proof: We start by removing vertices of degree one and zero, and contracting all vertices of degree two to a neighbor. This gives an equivalent instance with all vertices of degree at least three.

Erdös and Posa have shown in 1965 that there is a constant C, such that each graph has a feedback vertex set of size at most $Ck \log k$ or at least k vertex disjoint cycles. Now, we apply a 2-approximation algorithm for Feedback Vertex Set (e.g., from [3, 4]). If we obtain a feedback vertex set with $\geq 2Ck \log k$ vertices, then we can resolve the problem and answer 'yes'. Otherwise, let S be a feedback vertex set of size less than $2Ck \log k$, and let F be the forest $G - S$.

We build a collection $\mathbf{C}$ of disjoint paths between vertices in S. If there are ℓ disjoint paths between $s, s' \in S$, then these paths give us $\lfloor \ell/2 \rfloor$ disjoint cycles. So, if we have $(2Ck \log k)^2 + 2k$ disjoint paths in $\mathbf{C}$, we have k disjoint cycles and can decide 'yes'. Each leaf x in F must be incident to two vertices in S (as its degree in G is three), and we take a path between these two neighbors in S with x as only other vertex in $\mathbf{C}$. Thus, we can assume there are $O(k^2 \log^2 k)$ leaves in F, and hence also $O(k^2 \log^2 k)$ vertices in F with at least three neighbors in F. If F contains a path P with ℓ vertices, each incident to exactly two vertices, then as each vertex in P is incident to a vertex in S, such a path gives us $\lfloor \ell/2 \rfloor$ paths in $\mathbf{C}$. As the number of such paths is bounded by the number of vertices in F with at least three neighbors in F, we get a bound of $O(k^2 \log^2 k)$ on the number of vertices on these paths.

The construction shows that if F contains at least $5 \cdot (2Ck \log k)^2 + 2k$ vertices, there are k disjoint cycles, and thus we have a kernel of size $O(k^2 \log^2 k)$. $\square$

Several of these problems have been investigated for planar graphs. Here, polynomial and often linear size kernels exist, see [16, 7, 18]. We also consider the following well known problem, also known as k-Linkage.

Disjoint Paths
Input: Undirected graph $G = (V, E)$, vertices $s_1, \ldots, s_k, t_1, \ldots, t_k \in V$
Parameter: k
Question: Is there a collection of k paths $P_1, \ldots, P_k$ that are vertex disjoint, such that P_i is a path from s_i to t_i?

The result that the Disjoint Paths problem is fixed parameter tractable is a famous result by Robertson and Seymour as part of their fundamental work on graph minors: in [19], they show that for each fixed k, the problem can be solved in $O(n^3)$ time. Here, we give evidence that the problem has no kernel of polynomial size.

It is well known that the derived classic variants of Disjoint Cycles and Disjoint Paths are NP-complete, see [13, 15].

We introduce a new problem, which is used as an intermediate problem: we show that it is or-compositional and (its derived classic variant) NP-complete, and then give a polynomial time and parameter transformation to Disjoint Cycles, respectively Disjoint Paths.

Let L_k be the alphabet consisting of the letters $\{1, 2, \ldots, k\}$. We denote by L_k^* the set of words on L_k. A *factor* of a word $w_1 \cdots w_r \in L_k^*$ is a substring $w_i \cdots w_j \in L_k^*$, with $1 \le i < j \le r$, which starts and ends with the same letter, i.e., the factor has length at least two and $w_i = w_j$.

A word $W \in L_k^*$ has the *disjoint factor property* if one can find disjoint factors $F_1, \ldots, F_k$ in W such that the factor F_i starts and ends by the letter i. Observe that the difficulty lies in the fact that the factors F_i do not necessarily appear in increasing order, otherwise detecting them would be obviously computable in $O(n)$, where n is the length of W. We now introduce the parameterized problem Disjoint Factors.

The input of the Disjoint Factors problem is an integer $k \ge 1$ and a word W of L_k^*. The output is true if W has the disjoint factor property, otherwise false. This problem is clearly FPT since one can try all the $k!$ possible orders of the F_i's, and compute each of them linearly. A slightly more involved analysis gives an $O(nk \cdot 2^k)$ algorithm.

Proposition 17 *The* Disjoint Factors *problem can be solved in $O(nk \cdot 2^k)$ time.*

Proof: We use dynamic programming. Suppose word $W \in L_k^*$ is given, $|W| = n$. Write $W = w_1 w_2 \cdots w_n$. For each i, $0 \le i \le n$, and $S \subseteq L_k$, $x \in L_k$, let $A(S, i)$ and $B(S, x, i)$ be Boolean values, with $A(S, i)$ true, if and only if $w_1 \cdots w_i$ contains disjoint factors F_j for all $j \in S$, each F_j of length at least two and starting and ending with letter j. $B(S, x, i)$ is true, if and only if $w_1 \cdots w_i$ contains disjoint factors F_j for all $j \in S$, as above, and also a factor F_x, with F_x starting with the letter x and ending with w_i, and F_x also disjoint from the F_j, $j \in S$. I.e., the factor F_x ends at the substring $w_1 \cdots w_i$ in consideration.

It is a simple exercise in dynamic programming to give recurrences for A and B, and build a dynamic programming algorithm upon these that solves Disjoint Factors in $O(nk \cdot 2^k)$ time. $\square$

Theorem 18 *The* Disjoint Factors *is NP-complete.*

Proof: Clearly, the problem belongs to NP. We show NP-hardness by a transformation from 3-satisfiability. Let F be a 3-SAT formula with $c + 1$ clauses $C_0, \ldots, C_c$. We start our construction of our word W with the prefix $123123123456456456 \ldots (3c+1)(3c+2)(3c+3)(3c+1)(3c+2)(3c+3)(3c+1)(3c+2)(3c+3)$. Here the factor $(3i+1)(3i+2)(3i+3)(3i+1)(3i+2)(3i+3)(3i+1)(3i+2)(3i+3)$ corresponds to the clause C_i, for all $i = 0, \ldots, c$.

Note that the factor 123123123 does not have the disjoint factor property, but fails it only by one. Indeed, one can find two disjoint factors $\{F_1, F_2\}$, or $\{F_1, F_3\}$, etc, but not three disjoint factors F_1, F_2, F_3. Hence, in this prefix of W, one can find two disjoint factors for each clause, but not three.

Each variable appearing in F is a letter of our alphabet. In addition to W, we concatenate for each variable x a factor to W of the form $x p_1 p_1 p_2 p_2 \ldots p_l p_l x q_1 q_1 \ldots q_m q_m x$ where the p_i's are the position in which x appears as a positive literal, and the q_i's are the position in which x appears as a negative literal.

We feel that an example will clarify our discourse. To the formula $F = (\overline{x} \vee \overline{y} \vee z) \wedge (y \vee \overline{x} \vee \overline{z}) \wedge (x \vee y \vee z)$, we associate the word W_F
$123123123456456456789789789x77x1155xy4488y22yz3399z66z$

Observe that the solution $x = 1, y = 0, z = 0$ which satisfies F corresponds the the disjoint factors appearing in this order in W_F: 1231, 3123, 4564, 5645, 8978, 9789, 77, x1155x, y4488y, 22, z3399z, 66.

Finally, F is satisfiable if and only if W_F has the disjoint factor property. This proves our result. □

Lemma 19 Disjoint Factors *is or-composable.*

Proof: Suppose a collection of inputs $(W_1, k), \ldots, (W_t, k)$ for Disjoint Factors is given.

First we look at the case that $t > 2^k$. In this case, we solve each instance by the dynamic programming algorithm of Proposition 17. Note that the time to do this is polynomial in $\sum_1^t |W_i| + k$, as $2^k < \sum_1^t |W_i|$ here. So, we completely solve the problem, and can then transform to a trivial $O(1)$-size yes- or no-instance.

Now, suppose $t \leq 2^k$. We can assume that t is a power of two, say $t = 2^\ell$; if necessary, we add trivial no-instances ($k > 0$, W the empty string). For $1 \leq i \leq t$, $0 \leq j < \ell$, $i + 2^{j+1} \leq t + 1$, we define the word $W_{i,j}$ recursively as follows. If $j = 0$, $W_{i,0}$ is the word $(k+1)W_i(k+1)W_{i+1}(k+1)$. If $j > 0$, $W_{i,j}$ is the word $(k+1+j)W_{i,j-1}(k+1+j)W_{i+2^j,j-1}(k+1+j)$.

Note that $W_{i,j}$ contains each of the instances $W_i, \ldots, W_{i+2^{j+1}-1}$ as substrings. As result of the composition, we take the word $W' = W_{1,\ell-1}$.

In other words, W' is the limit word of

- $(k+1)W_1(k+1)W_2(k+1)$
- $(k+2)(k+1)W_1(k+1)W_2(k+1)(k+2)(k+1)W_3(k+1)W_4(k+1)(k+2)$
- $(k+3)(k+2)(k+1)W_1(k+1)W_2(k+1)(k+2)(k+1)W_3(k+1)W_4(k+1)(k+2)(k+3)(k+2)(k+1)W_5(k+1)W_6(k+1)(k+2)(k+1)W_7(k+1)W_8(k+1)(k+2)(k+3)$
  ...

The resulting instance is $(W', k + \ell)$.

By construction, (W', k') has a solution, if and only if at least one of the (W_i, k) has a solution. Suppose (W', k') has a solution. Then, there are two possibilities for the factor $F_{k+\ell}$: either it is $(k+\ell)W_{1,\ell-2}(k+\ell)$, or $(k+\ell)W_{1+2^{\ell-1},\ell-2}(k+\ell)$. In the first case, it 'shields' the instances $W_1, \ldots, W_{2^{\ell-1}}$; in the other case, it 'shields' the instances $W_{1+2^\ell-1}, \ldots, W_{2^\ell}$. No other factors can be taken in the shielded part. This repeats with the other symbols above k: $F_{k+\ell-1}$ shields half of what was left by $F_{k+\ell}$, and one can see that there remains exactly one substring W_i that does not belong to any of the F_i with $i > k$. In this substring, we must find the factors $F_1, \ldots, F_k$, and thus there is at least one (W_i, k) which has a solution.

Suppose (W_i, k) has a solution. We take from W' the factors $F_1, \ldots, F_k$ from W_i. The other factors can be easily chosen: take factors $F_{k+\ell}$, $F_{k+\ell-1}$, etc., in this order, each time taking the unique possibility which does not overlap already chosen factors.

Finally, note that $k + \ell \leq 2k$, so we have a polynomial time and parameter transformation. □

Corollary 20 DISJOINT FACTORS *belongs to the class NPK^0_{or}, and thus has no polynomial kernel, unless $NP \subseteq coNP/poly$.*

We now show that DISJOINT CYCLES belongs to NPK_{or} (and hence has no polynomial kernel, unless $NP \subseteq coNP/poly$), by giving a polynomial time and parameter transformation from DISJOINT FACTORS to DISJOINT CYCLES.

Theorem 21 DISJOINT CYCLES $\in NPK_{or}$*, and hence has no polynomial kernel unless $NP \subseteq coNP/poly$.*

Proof: We use the following polynomial time and parameter transformation from DISJOINT FACTORS. Given an input (W, k) of DISJOINT FACTORS, with $W = w_1 \cdots w_n$ a word in L_k^*, we build a graph $G = (V, E)$ as follows. First, we take n vertices $v_1, \ldots, v_n$, and edges $\{v_i, v_{i+1}\}$ for $1 \leq i < n$, i.e., these vertices form a path of length n. Call this subgraph of G P. Then, for each $i \in L_k$, we add a vertex x_i, and make x_i incident to each vertex v_j with $w_j = i$, i.e., to each vertex representing the letter i.

G has k disjoint cycles, if and only if (W, k) has the requested k disjoint factors.

Suppose G has k disjoint cycles $c_1, \ldots, c_k$. As P is a path, each of these cycles must contain at least one vertex not on P, i.e., of the form x_j, and hence each of these cycles contains exactly one vertex x_j. For $1 \leq j \leq k$, the cycle c_j thus consists of x_j and a subpath of P. This subpath must start and end with a vertex incident to x_j. These both represent letters in W equal to j. Let F_j be the factor of W corresponding to the vertices on P in c_j. Now, $F_1, \ldots, F_k$ are disjoint factors, each of length at least two (as the cycles have length at least three), and F_j starts and ends with j, for all j, $1 \leq j \leq k$.

Conversely, if we have disjoint factors $F_1, \ldots, F_k$ with the properties as in the DISJOINT FACTORS problem, we build k vertex disjoint cycles as follows: for each j, $1 \leq j \leq k$, take the cycle consisting of x_j and the vertices corresponding to factor F_j.

As DISJOINT CYCLES in an NP-complete problem, the transformation just given and the membership of DISJOINT FACTORS in NPK^0_{or} show that DISJOINT CYCLES $\in NPK_{or}$.

By Corollary 11, it follows that DISJOINT CYCLES has no polynomial kernel unless $NP \subseteq coNP/poly$. □

A simple modification of the proof above gives our desired result for DISJOINT PATHS.

Theorem 22 DISJOINT PATHS $\in NPK_{or}$*, and hence has no polynomial kernel unless $NP \subseteq coNP/poly$.*

Proof: Suppose we have input (W, k) for the Disjoint Factors problem, W a word in L_k^*. Build a graph G as follows. Start with a path P with vertices $v_1, \ldots, v_n$ (as in the previous proof). For each $i \in L_k$, take a new vertex x_i and a new vertex y_i. Make x_i incident to each vertex representing the first, third, fifth, etc., occurrence of the letter i in W, and make y_i incident to each vertex representing the second, fourth, sixth, etc., occurrence of the letter i in W.

Now, the Disjoint Factors problem has a solution, if and only if it has a solution where each factor F_i starts and ends with an i but has no other i (otherwise we can replace the solution by an equivalent one where F_i is shorter), and hence is between an even and an odd occurrence of the letter i. With a proof, similar to the proof of Theorem 21, correctness of the construction follows. □

4.2 Problems parameterized by treewidth

We now give a result with an easy proof, but one that seems not to be obtainable with only the techniques from [6]. Consider the following two problems.

Hamiltonian Path, with given endpoints and given tree decomposition
Input: Undirected graph $G = (V, E)$, vertices $s, t \in V$, tree decomposition of G.
Parameter: Width of the tree decomposition
Question: Does G have an Hamiltonian Path that starts in s and ends in t?

Hamiltonian Circuit with given tree decomposition
Input: Undirected graph $G = (V, E)$, tree decomposition of G.
Parameter: Width of the tree decomposition
Question: Does G have a Hamiltonian Circuit?

Theorem 23 Hamiltonian Path, with given endpoints and given tree decomposition *and* Hamiltonian Circuit with given tree decomposition *belong to the class NPK_{and}.*

Proof: It is well known that the classic version of the problems are NP-complete. Hamiltonian Path, with given endpoints and given tree decomposition is and-composable: if we have a series of r instances, we take the disjoint union and identify the vertex playing the role of t in instance i with the vertex playing the role of s in instance $i+1$ $(1 \leq i \leq r-1.)$ Building a tree decomposition can be done as follows: first, take the disjoint union of the tree decompositions of the r instances. For each i, $1 \leq i \leq r-1$, take a bag i_t in the tree decomposition of the ith instance that contains the vertex playing the role of t in that instance, and take a bag j_s in the tree decomposition of the $(i+1)$st instance that contains the vertex playing the role of s in that instance, and add the edge from bag i_t to j_s. It is easy to verify that this gives the desired tree decomposition. Thus, we can conclude that HAMILTONIAN PATH, WITH GIVEN ENDPOINTS AND GIVEN TREE DECOMPOSITION belongs to $NPK^0_{and} \subseteq NPK_{and}$.

The well known transformation from HAMILTONIAN PATH to HAMILTONIAN CIRCUIT can now be used here as well: add one new vertex to the graph making it adjacent to s and t. A tree decomposition of the new graph whose width is one larger can be made by adding the new vertex to each bag. Thus, we have a polynomial time and parameter transformation. This shows that HAMILTONIAN CIRCUIT WITH GIVEN TREE DECOMPOSITION belongs to NPK_{and}. □

Using this result, we can also show that HAMILTONIAN PATH (with or without specified endpoints) and HAMILTONIAN CIRCUIT do not have kernels polynomial in the treewidth of the graph, unless the and-distillation conjecture does not hold, i.e., without assuming that a tree decomposition is given as part of the input. This can be shown by using that there is a $O(\log k)$-approximation for treewidth, see [2, 1] or [8], where k is the treewidth.

We expect that with similar techniques, many similar results can be obtained for graph problems, parameterized by treewidth or other notions like branchwidth, pathwidth, cliquewidth, ...

5 Conclusions

In this paper, we introduced a new technique to give evidence that specific combinatorial problems do not have a kernel of polynomial size. The technique mimics the way we are used to give evidence that problems do not have a polynomial time algorithm: instead of polynomial time transformations to an NP-complete problem, we now use polynomial time and parameter transformations to problems in the classes NPK_{and} or NPK_{or}. In this way, we make a contribution to the toolbox for a theoretical analysis what can be achieved by preprocessing for combinatorial problems.

In addition, we showed for some problems that they do not have a kernel of polynomial size, i.e., we cannot preprocess the problems such that the resulting instances have a size bounded by a polynomial of the specific parameter, unless certain complexity theoretic conjectures do not hold. The results for DISJOINT CYCLES and DISJOINT PATHS are most interesting, and were to us unexpected, as several closely related and similar problems do have kernels of polynomial sizes. Thus, the result in our paper for e.g. DISJOINT CYCLES plays the role that it can direct further research efforts, i.e., it appears not to be useful to aim at finding a polynomial kernel for the problem; this is somewhat comparable to stating that an NP-completeness proof directs our research efforts away from finding a polynomial time algorithm for a problem.

The further development of the theory of kernel sizes is an interesting topic for further research. An important topic with many recent results (see e.g., the overview paper by Guo and Niedermeier [14]) is to find kernels of sizes as small as possible for concrete combinatorial problems. Some questions we want to add to this are: The theory so far allows to distinguish between constant size, polynomial size, and any size kernels: can we refine this? Are the classes NPK_{and} and NPK_{or} the same? (There are some problems that belong to both classes.) Are there complete or "hardest" problems for these classes? Another still open problem is whether there exists a result for and-distillation that is similar to the result of Fortnow and Santhaman for the or-distillation conjecture, i.e., can we relate the and-distillation conjecture to more widely known and believed complexity theoretic conjectures?

References

[1] E. Amir. Approximation algorithms for treewidth. To appear in Algorithmica, 2008.

[2] E. Amir. Efficient approximation for triangulation of minimum treewidth. In *Proceedings of the 17th Conference on Uncertainty in Artificial Intelligence*, pages 7–15, 2001.

[3] V. Bafna, P. Berman, and T. Fujito. A 2-approximation algorithm for the undirected feedback vertex set problem. *SIAM J. Disc. Math.*, 12:289–297, 1999.

[4] A. Becker and D. Geiger. Optimization of Pearl's method of conditioning and greedy-like approximation algorithms for the vertex feedback set problem. *Artificial Intelligence*, 83:167–188, 1996.

[5] H. L. Bodlaender. A cubic kernel for feedback vertex set. In W. Thomas and P. Well, editors, *Proceedings 24th International Symposium on Theoretical Aspects of Computer Science, STACS 2007*, pages 320–331. Springer Verlag, Lecture Notes in Computer Science, vol. 4393, 2007.

[6] H. L. Bodlaender, R. G. Downey, M. R. Fellows, and D. Hermelin. On problems without polynomial kernels (extended abstract). In L. Aceto, I. Damgård, L. A. Goldberg, M. M. Halldórsson, A. Ingólfsdóttir, and I. Walukuewics, editors, *Proceedings of the 34th International Colloquium on Automata, Languages and Programming, ICALP 2008*, pages 563–574. Springer Verlag, Lecture Notes in Computer Science, vol. 5125, 2008.

[7] H. L. Bodlaender and E. Penninkx. A linear kernel for planar feedback vertex set. In M. Grohe and R. Niedermeier, editors, *Proceedings Third International Workshop on Parameterized and Exact Computation, IWPEC 2008*, pages 160–171. Springer Verlag, Lecture Notes in Computer Science, vol. 5018, 2008.

[8] V. Bouchitté, D. Kratsch, H. Müller, and I. Todinca. On treewidth approximations. *Disc. Appl. Math.*, 136:183–196, 2004.

[9] R. G. Downey and M. R. Fellows. Fixed-parameter tractability and completeness I: Basic results. *SIAM J. Comput.*, 24:873–921, 1995.

[10] R. G. Downey and M. R. Fellows. Fixed-parameter tractability and completeness II: On completeness for $w[1]$. *Theor. Comp. Sc.*, 141:109–131, 1995.

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

[12] L. Fortnow and R. Santhanam. Infeasibility of instance compression and succinct PCPs for NP. In *Proceedings of the 38th Annual Symposium on Theory of Computing, STOC 2006*, pages 133–142. ACM Press, 2008.

[13] M. R. Garey and D. S. Johnson. *Computers and Intractability, A Guide to the Theory of NP-Completeness.* W.H. Freeman and Company, New York, 1979.

[14] J. Guo and R. Niedermeier. Invitation to data reduction and problem kernelization. *ACM SIGACT News*, 38:31–45, 2007.

[15] R. M. Karp. On the complexity of combinatorial problems. *Networks*, 5:45–68, 1975.

[16] T. Kloks, C. M. Lee, and J. Liu. New algorithms for k-face cover, k-feedback vertex set, and k-disjoint cycles on plane and planar graphs. In *Proceedings 28th International Workshop on Graph-Theoretic Concepts in Computer Science WG'02*, pages 282–295. Springer Verlag, Lecture Notes in Computer Science, vol. 2573, 2002.

[17] L. Mathieson, E. Prieto, and P. Shaw. Packing edge disjoint triangles: A parameterized view. In R. G. Downey and M. R. Fellows, editors, *Proceedings 1st International Workshop on Parameterized and Exact Computation, IWPEC 2004*, pages 127–137. Springer Verlag, Lecture Notes in Computer Science, vol. 3162, 2004.

[18] E. Penninkx and H. L. Bodlaender. A linear kernel for the k-disjoint cycles problem on planar graphs. To appear in proceedings ISAAC 2008, 2008.

[19] N. Robertson and P. D. Seymour. Graph minors. XIII. The disjoint paths problem. *J. Comb. Theory Series B*, 63:65–110, 1995.

[20] S. Thomassé. A quadratic kernel for feedback vertex set. Unpublished manuscript. To appear in Proceedings SODA 2009, 2008.