Data-Compression for Parametrized Counting Problems on Sparse Graphs

Eun Jung Kim, Maria Serna, Dimitrios M. Thilikos

Data-Compression for Parametrized Counting Problems on Sparse Graphs

Eun Jung Kim[1]
Université Paris-Dauphine, PSL Research University, CNRS/LAMSADE, 75016, Paris, France
eun-jung.kim@dauphine.fr

Maria Serna[2]
Computer Science Department & BGSMath, Universitat Politècnica de Catalunya, Barcelona, Spain
mjserna@cs.upc.edu

Dimitrios M. Thilikos[3]
AlGCo project-team, LIRMM, Université de Montpellier, CNRS, Montpellier, France; and Department of Mathematics, National and Kapodistrian University of Athens, Greece
sedthilk@thilikos.info

Abstract

We study the concept of *compactor*, which may be seen as a counting-analogue of kernelization in counting parameterized complexity. For a function $F : \Sigma^* \to \mathbb{N}$ and a parameterization $\kappa : \Sigma^* \to \mathbb{N}$, a compactor (P, M) consists of a polynomial-time computable function P, called *condenser*, and a computable function M, called *extractor*, such that $F = \mathsf{M} \circ \mathsf{P}$, and the condensing $\mathsf{P}(x)$ of x has length at most $s(\kappa(x))$, for any input $x \in \Sigma^*$. If s is a polynomial function, then the compactor is said to be of polynomial-size. Although the study on counting-analogue of kernelization is not unprecedented, it has received little attention so far. We study a family of vertex-certified counting problems on graphs that are MSOL-expressible; that is, for an MSOL-formula ϕ with one free set variable to be interpreted as a vertex subset, we want to count all $A \subseteq V(G)$ where $|A| = k$ and $(G, A) \models \phi$. In this paper, we prove that every vertex-certified counting problems on graphs that is *MSOL-expressible* and *treewidth modulable*, when parameterized by k, admits a polynomial-size compactor on H-topological-minor-free graphs with condensing time $O(k^2n^2)$ and decoding time $2^{O(k)}$. This implies the existence of an FPT-algorithm of running time $O(n^2k^2) + 2^{O(k)}$. All aforementioned complexities are under the Uniform Cost Measure (UCM) model where numbers can be stored in constant space and arithmetic operations can be done in constant time.

2012 ACM Subject Classification Theory of computation → Graph algorithms analysis

Keywords and phrases Parameterized counting, compactor, protrusion decomposition

Digital Object Identifier 10.4230/LIPIcs.ISAAC.2018.20

Related Version The full version of this extended abstract has appeared in [35], `http://arxiv.org/abs/1809.08160`.

[1] Supported by project ESIGMA (ANR-17-CE23-0010).
[2] Partially funded by MINECO and FEDER funds under grants TIN2017-86727-C2-1-R (GRAMM) and MDM-2014-044 (BGSMath), and by AGAUR grant 2017SGR-786 (ALBCOM).
[3] Supported by projects DEMOGRAPH (ANR-16-CE40-0028) and ESIGMA (ANR-17-CE23-0010).

1 Introduction

A large part of research on parameterized algorithms has been focused on algorithmic techniques for parametrizations of decision problems. However, relatively less effort has been invested for solving parameterized counting problems. In this paper, we provide a general data-reduction concept for counting problems, leading to a formal definition of the notion of a *compactor*. Our main result is an algorithmic meta-theorem for the existence of a polynomial size compactor, that is applicable to a wide family of problems of graphs.

1.1 General context

Algorithmic meta-theorems. Parameterized complexity has been proposed as a multivariable framework for coping with the inherent complexity of computational problems . Nowadays, it is a mature discipline of modern Theoretical Computer Science and has offered a wealth of algorithmic techniques and solutions (see [12, 17, 22, 38] for related textbooks). In some cases, in-depth investigations on the common characteristics of parameterized problems gave rise to algorithmic meta-theorems. Such theorems typically provide conditions, logical and/or combinatorial, for a problem to admit a parameterized algorithm [30, 29, 36, 40]. Important algorithmic meta-theorems concern model-checking for Monadic Second Order Logic (MSOL) [10, 7, 2, 41] on bounded treewidth graphs and model checking for First Order Logic (FOL) on certain classes of sparse graphs [21, 28, 13, 20, 19, 31].

In some cases, such theorems have a counterpart on *counting* parameterized problems. Here the target is to prove that counting *how many solutions* exist for a problem is fixed parameter tractable, under some parameterization of it. Related meta-algorithmic results concern counting analogues of Courcelle's theorem, proved in [9], stating that counting problems definable in MSOL are fixed-parameter tractable when parameterized by the treewidth of the input graph. Also similar results for certain fragments of MSOL hold when parameterized by the rank-width of the input graph [9]. Moreover, it was shown in [27] that counting problems definable in first-order logic are fixed-parameter tractable on locally tree-decomposable graphs (e.g. for planar graphs and bounded genus graphs).

Kernelization and data-reduction. A well-studied concept in parameterized complexity is kernelization. We say that a parameterized problem admits a polynomial kernel if there is an algorithm – called *kernelization algorithm* – that can transform, in polynomial time, every input instance of the problem to an equivalent one, whose size is bounded by a function of the parameter. When this function is polynomial then we have a *polynomial kernel*. A polynomial kernel permits the drastic data-reduction of the problem instances to equivalent "miniatures" whose size is *independent* from the bulk of the input size and is polynomial on the parameter. That way, a polynomial kernel, provides a preprocessing of computationally hard problems that enables the application of exact algorithmic approaches (however still super-polynomial) on significantly reduced instances [37].

Meta-algorithmic results for kernelization. Apart from the numerous advances on the design of polynomial kernels for particular problems, algorithmic meta-theorems appeared also for kernelization. The first result of this type appeared in [4], where it was proved that certain families of problems on graphs admit polynomial kernels on bounded genus graphs. The logic-condition of [4] is CMSOL-expressibility or, additionally, the Finite Integer Index (FII) property (see [1, 6, 14]). Moreover, the meta-algorithmic results of [4] require additional combinatorial properties for the problems in question. The results in [4] where

extended in [23] (see also [25]) where the combinatorial condition for the problem was related to bidimensionality, while the applicability of the results was extended in minor-closed graph classes. Finally, further extensions appeared in [34] where, under the bounded treewidth-modulability property (see Subsection 1.2), some of the results in [23, 4] could be applied to more graph classes, in particular those excluding some fixed graph as a topological minor.

Data reduction for counting problems. Unfortunately, not much has been done so far in the direction of data-reduction for parameterized counting problems. The most comprehensive work in this direction was done by Marc Thurley [42] (see also [43]) who proposed the first formal definition of a kernelization analogue for parameterized problems called *counting kernelization*. In [42] Thurley investigated up to which extent classic kernelization techniques such as *Buss' Kernelization* and *crown decomposition* may lead to counting counterparts of kernelization. In this direction, he provided counting kernelizations for a series of parameterized counting problems such as and p-#VertexCover, p-card-#Hitting Set and p-#Unique Hitting Set.

Compactor enumeration. Another framework for data-reduction on parameterized counting problems is provided by the notion of a *compactor*. In a precursory level, it appeared for the first time in [16]. The rough idea in [16] was to transform the input of a parameterized counting problem to a structure, called *the compactor*, whose size is (polynomially) bounded by the parameter and such that the enumeration of certain family of objects (referred as *compactor enumeration* in [16]) in the compactor is able to derive the number of solutions for the initial instance. This technique was introduced in [16] for counting restrictive list H-colorings and, later in [39], for counting generalized coverings and matchings. However none of [16, 39] provided a general formal definition of a compactor, while, in our opinion, the work of Thurley provides a legitimate formalization of compactor enumeration.

In this paper, we define formally the concept of a compactor for parameterizations of function problems (that naturally include counting problems) that is not based on enumeration. As a first step, we observe that for parameterized function problems, the existence of a compactor is equivalent to the existence of an FPT-algorithm, a fact that is also the case for classic kernels on decision problems and for counting kernels in [42].

Under the above formal framework, we prove an algorithmic meta-theorem on the existence of polynomial compactors for a general family of graph problems. In the next subsection, we define the compactor concept and we present the related meta-algorithmic results.

1.2 Our results

Counting problems and parameterizations. First of all notice that, for a counting problem, it is not possible to have a kernelization in the classic sense, that is to produce an reduced instance, bounded by a function of k, that is counting-equivalent in the sense that the number of solutions in the reduced instance will provide the number of solutions in the original one. For this reason we need a more refined notion of data compression where we transform the input instance to "structure", whose size is bounded by a function of k. This structure contains enough information (combinatorial and arithmetical) so as to permit the recovering of the number of the solutions in the initial instance. We next formalize this idea to the concept of a compactor.

Let $\mathbb{N}$ be all non-negative integers and by poly the set of all polynomials. Let Σ be a fixed alphabet. A *parameterized function problem* is a pair (F, κ) where $F, \kappa : \Sigma^* \to \mathbb{N}$. An FPT-algorithm for (F, κ) is one that, given $x \in \Sigma^*$, outputs $F(x)$ in $f(\kappa(x)) \cdot \mathsf{poly}(|x|)$

steps. When evaluating the running time, we use the standard Uniform Cost Measure (UCM) model where all basic arithmetic computations are carried out in constant time. We also disregard the size of the numbers that are produced during the execution of the algorithm.

Compactors. Let (F, κ) be a parameterized function problem. A *compactor* for (F, κ) is a pair (P, M) where

- $P : \Sigma^* \to \Sigma^*$ is a polynomially computable function, called an *condenser*,
- $M : \Sigma^* \to \mathbb{N}$ is a computable function, called a *extractor*,
- $F = M \circ P$, i.e., $\forall x \in \Sigma^*$, $F(x) = (M \circ P)(x)$, and
- there is a recursive function $s : \mathbb{N} \to \mathbb{N}$ where $\forall x \in \Sigma^*$ $|P(x)| \leq s(\kappa(x))$.

We call the function s *size* of the compactor (P, M) and, if $s \in \mathsf{poly}$, we say that (P, M) is a *polynomial-size compactor* for (F, κ). We call the running time of the algorithm computing P, measured as a function of $|x|$, *condensing time* of (P, M). We also call the running time of the algorithm computing M, measured as a function of $\kappa(x)$, *decoding time* of (P, M). We can readily observe that parameterized function problem has an FPT-algorithm if and only if there is a compactor for it.

Up to our best knowledge, the notion of compactor as formalized in this paper is new. As discussed in Subsection 1.1, similar notions have been proposed such as counting kernelization [42] and compactor enumeration [16]. In both counting kernelization and compact enumeration, a mapping from the set of all certificates to certain objects in the new instance is required. While this approach comply more with the idea of classic kernelization, it seems to be more restrictive. The main difference of our compactor from the previous notions is that (the condenser of) a compactor is free of this requirement, which makes the definition more flexible and easier to work with. Due to this flexibility and succinctness, we believe that our notion might be amenable for lower bound machineries akin to those for decision problem kernelizations.

Parameterized counting problems on graphs. A *structure* is a pair (G, A) where G is a graph and $A \subseteq V(G)$. Given a MSOL-formula ϕ on structures and some graph class $\mathcal{G}$, we consider the following parameterized counting problem $\Pi_{\phi,\mathcal{G}}$.

$\Pi_{\phi,\mathcal{G}}$
Input: a graph $G \in \mathcal{G}$, an non-negative integer k.
Parameter: k.
Count: the number of vertex sets $A \subseteq V(G)$ such that $(G, A) \models \phi$ and $\lvert A\rvert = k$.

We say that an instance $(G, k) \in \mathcal{G} \times \mathbb{N}$ of $\Pi_{\phi,\mathcal{G}}$ is a *null* instance if it has no solutions. Given a graph G, we say that a vertex set $A \subseteq V(G)$ is a *t-treewidth modulator* of G if the removal of A from G leaves a graph of treewidth at most t. Given an MSOL-formula ϕ and a graph class $\mathcal{G}$, we say that $\Pi_{\phi,\mathcal{G}}$ is *treewidth modulable* if there is a constant t (depending on ϕ and $\mathcal{G}$ only) such that, for every non-null instance (G, k) of $\Pi_{\phi,\mathcal{G}}$, G has a t-treewidth modulator of size at most $t \cdot k$.

Let $\mathcal{F}_H$ be the class of all graphs that do not contain a subdivision of H as a subgraph. The next theorem states our main result.

▶ **Theorem 1.** *For every graph H and every MSOL-formula ϕ, if $\Pi_{\phi,\mathcal{F}_H}$ is treewidth modulable, then there is a compactor for $\Pi_{\phi,\mathcal{F}_H}$ of size $O(k^2)$ with condensing time $O(k^2n^2)$ and decoding time $2^{O(k)}$.*

As a corollary of the main theorem we have the following.

▶ **Corollary 2.** *For every graph H and every MSOL-formula ϕ, if $\Pi_{\phi,\mathcal{F}_H}$ is treewidth modulable, then $\Pi_{\phi,\mathcal{F}_H}$ can be solved in $O(k^2n^2) + 2^{O(k)}$ steps.*

In the above results, the constants hidden in the O-notation depend on the choice of ϕ, on the treewidth-modulability constant t, and on the choice of H.

Recall that the above results are stated using the UCM model. As for $\Pi_{\phi,\mathcal{F}_H}$, the number of solutions is $O(n^k)$ and this number can be encoded in $O(k \log n)$ bits. Assuming that summations of two r-bit numbers can be done in $O(r)$ steps and multiplications of two r-bit numbers can be done in $O(r^2)$ steps, then the size of the compactor in Theorem 1 is $O(k^2 \log n)$ the condensing and extracting times are $O(k^4n^2 \log^2 n)$ and $2^{O(k)} \log^2 n$ respectively. Consequently, the running time of the algorithm in Corollary 2 is $O(k^4n^2 \log^2 n) + 2^{O(k)} \log^2 n$.

Coming back to the algorithmic meta-theorems on parameterized counting problems we should remark that the problem condition of Corollary 2 is weaker than MSOL, as it additionally demands treewidth-modulability. However, the graph classes where this result applies have unbounded treewidth or rankwidth. That way our results can be seen as orthogonal to those of [9].

On the side of FOL, the problem condition of Corollary 2 is stronger than FOL, while its combinatorial applicability includes planar graphs or graphs of bounded genus where, the existing algorithmic meta-theorems require FOL-expressibility (see [27]).

1.3 Outline of the compactor algorithms

Our approach follows the idea of applying data-reduction based on protrusion decomposability. This idea was initiated in [4] for the automated derivation of polynomial kernels on decision problems. The key-concept in [4] is the notion of a *protrusion*, a set of vertices with small neighborhood to the rest of the graph and inducing a graph of small treewidth. Also, [4] introduced the notion of a *protrusion decomposition*, which is a partition of G to $O(k)$ graphs such the first one is a "center", of size $O(k)$, and the rest are protrusions whose neighborhoods are in the center.

The meta-algorithmic machinery of [4] is based on the following combinatorial fact: for the problems in question, YES-instances – in our case non-null instances– admit a protrusion decomposition that, when the input has size $\Omega(k)$, one of its protrusions is "big enough". This permits the application of some "graph surgery" that consists in replacing a big protrusion with a smaller one and, that way, creates an equivalent instance of the problem (the replacements are based on the MSOL-expressibiliy of the problem). In the case of counting problems, this protrusion replacement machinery does not work (at least straightforwardly) as we have to keep track, not only of the way some part of a solution "invades" a protrusion, but also of the number of *all* those partial solutions. Instead, we take another way that avoids stepwise protrusion replacement. In our approach, the condenser of the compactor first constructs an approximate protrusion decomposition, then, it computes how many possible partial solutions of all possible sizes may exist in each one of the protrusions. This computation is done by dynamic programming (see Section 4) and produces a total set of $O(k^2)$ arithmetic values. These values, along with the combinatorial information of the center of the protrusion decomposition and the neighborhoods of the protrusions in the center, constitutes the output of the condenser. This structure can be stored in $O(k^2)$ space (given that arithmetic values can be stored in constant space) and contains enough information to obtain the number of all the solutions of the initial instance in $2^{O(k)}$ steps (Section 4).

We stress that the above machinery demands the polynomial-time construction of a constant-factor approximation of a protrusion-decomposition. To our knowledge, this remains an open problem in general. So far, no such algorithm has been proposed, even for particular

graph classes, mostly because meta-kernelization machinery in [4] (and later in [25, 23, 34, 24]) is based on stepwise protrusion replacement and does not actually need to construct such a decomposition. Based on the result in [34], we show that that the construction of such an approximate protrusion decomposition is possible on H-topological-minor-free graphs, given that it is possible to construct an approximate t-treewidth modulator of G. In fact, this can been done in general graphs using the randomized constant-factor approximation algorithm in [24]. Responding to the need for a deterministic approximation we provide a constant-factor approximation algorithm that finds a t-treewidth modulator on H-topological-minor free graphs (Section 3). This algorithm runs in $O(k^2n^2)$ steps and, besides from being a necessary step of the condenser of our compactor, is of independent algorithmic interest.

2 Preliminaries

We use $\mathbb{N}$ to denote the set of all non-negative integers. Let $\chi : \mathbb{N}^2 \to \mathbb{N}$ and $\psi : \mathbb{N} \to \mathbb{N}$. We say that $\chi(n,k) = O_k(\psi(n))$ if there exists a function $\phi : \mathbb{N} \to \mathbb{N}$ such that $\chi(n,k) = O(\phi(k) \cdot \psi(n))$. Given $a, b \in \mathbb{N}$, we define by $[a,b] = \{a, \ldots, b\}$. Also, given some $a \in \mathbb{N}$ we define $[a] = \{1, \ldots, a\}$. Given a set Z and a $k \in \mathbb{N}$, we denote $\binom{Z}{k} = \{S \subseteq Z \mid |S| = k\}$.

2.1 Graphs and boundary graphs

Graphs. All graphs in this paper are simple and undirected. Given a graph G, we use $V(G)$ to denote the set of its vertices. Given a $S \subseteq V(G)$ we denote by $N_G(S)$ the set of all neighbours of S in G that are not in S. We also set $N_G[S] = S \cup N_G(S)$ and we use $N(S)$ and $N[S]$ as shortcuts of $N_G(S)$ and $N_G[S]$ (when the index is a graph denoted by G). We define $G - S$ as the graph obtained from G if we remove the vertices in S, along with the edges incident to them. The *subgraph of G induced by S* is the graph $G[S] := G - (V(G) \setminus S)$. Finally, we set $\partial_G(S) = N_G(V(G - S))$. We call $|V(G)|$ the *size* of a graph G and n is reserved to denote the size of the input graph for time complexity analysis.

Given a graph G, a *subdivision* of G is any graph that is obtained from G after replacing its edges by paths with the same endpoints. We say that a graph H is a *topological minor* of G if G contains as a subgraph some subdivision of H. We also say that G is *H-topological-minor-free* if it excludes H as a topological minor.

Boundaried structures. A *labeling* of a graph G is any injective function $\lambda : V(G) \to \mathbb{N}$. Given a structure (G, A), we call A *the annotated set* of (G, A) and the vertices in A *annotated vertices* of (G, A).

A *boundaried structure*, in short a *b-structure*, is a triple $\mathbf{G} = (G, B, A)$ where G is a graph and $B, A \subseteq V(G)$. We say that B is the *boundary* of $\mathbf{G}$ and A is the *annotated set* of $\mathbf{G}$. Also we call the vertices of B *boundary vertices* and the vertices in A *annotated vertices*. We use notation $\mathcal{B}^{(t)}$ to denote all b-structures whose boundary has at most t vertices. We set $G(\mathbf{G}) = G$, $V(\mathbf{G}) = V(G)$, $B(\mathbf{G}) = B$, $A(\mathbf{G}) = A$. We refer to G as *the underlying graph* of $\mathbf{G}$ and we always assume that the underlying graph of a b-structure is accompanied with some labelling λ. Under the presence of such a labelling, we define the *index* of a boundary vertex v as the quantity $|\{u \in B \mid \lambda(u) \leq \lambda(v)\}|$ i.e., the index of v when we arrange the vertices of B according to λ in increasing order. We extend the notion of index to subsets of B in the natural way, i.e., the index of $S \subseteq B$ consists of the indices of all the vertices in S.

A *boundaried graph*, in short *b-graph*, is any b-structure $\mathbf{G} = (G, B, A)$ such that $A = V(G)$. For simplicity we use the notation $\mathbf{G} = (G, B, -)$ to denote b-graphs instead of using the heavier notation $\mathbf{G} = (G, B, V(G))$. For every $t \in \mathbb{N}$, we use $\overline{\mathcal{B}}^{(t)}$ to denote the

b-graphs in $\mathcal{B}^{(t)}$. We avoid denoting a boundary graph as an annotated graph as we want to stress the role of B as a boundary.

We say that two b-structures $\mathbf{G}_1 = (G_1, B_1, A_1)$ and $\mathbf{G}_2 = (G_2, B_2, A_2)$ are *compatible*, denoted by $\mathbf{G}_1 \sim \mathbf{G}_2$, if $A_1 \cap B_1$ and $A_2 \cap B_2$ have the same index and the labeled graphs $G[B_1]$ and $G[B_2]$, where each vertex of B_i is labeled by its index, are identical.

Given two compatible b-structures $\mathbf{G}_1 = (G_1, B_1, A_1)$ and $\mathbf{G}_2 = (G_2, B_2, A_2)$, we define $\mathbf{G}_1 \oplus \mathbf{G}_2$ as the structure (G, A) where

- the graph G is obtained by taking the disjoint union of G_1 and G_2 and then identifying boundary vertices of G_1 and G_2 of the same index, and
- the vertex set A is obtained from A_1 and A_2 after identifying equally-indexed vertices in $A_1 \cap B_1$ and $A_2 \cap B_2$.

Keep in mind that $(G, A) = \mathbf{G}_1 \oplus \mathbf{G}_2$ is an annotated graph and not a b-structure. We always assume that the labels of the boundary of $\mathbf{G}_1$ prevail during the gluing operation, i.e., they are inherited to the identified vertices in (G, A) while the labels of the boundary of $\mathbf{G}_2$ dissapear in (G, A). Especially, when $\mathbf{G}_1$ and $\mathbf{G}_2$ are compatible b-graphs, we treat $\mathbf{G}_1 \oplus \mathbf{G}_2$ as a graph for notational simplicity.

Treewith of b-structures. Given a b-structure $\mathbf{G} = (G, B, A)$, we say that the triple $D = (T, \chi, r)$ is a *tree decomposition* of $\mathbf{G}$ if (T, χ) is a tree decomposition of G, $r \in V(T)$, and $\chi(r) = B$. We see T as a tree rooted on r. The *width* of a tree decomposition $D = (T, \chi, r)$ is the width of the tree decomposition (T, χ). The treewidth of a b-structure $\mathbf{G}$ is the minimum width over all its tree decompositions and is denoted by $\mathbf{tw}(\mathbf{G})$. We use $\mathcal{T}^{(t)}$ (resp. $\overline{\mathcal{T}}^{(t)}$) to denote all b-structures (resp. b-graphs) in $\mathcal{B}^{(t)}$ (resp. $\overline{\mathcal{B}}^{(t)}$) with treewidth at most t.

Protrusion decompositions. Let G be a graph. Given $\alpha, \beta, \gamma \in \mathbb{N}$, an (α, β, γ)-*protrusion decomposition of* G is a sequence of $\mathbf{G}_1 = (G_1, B_1, -), \ldots, \mathbf{G}_s = (G_s, B_s, -)$ of b-graphs where, given that $X_i = V(G_i) \setminus B_i, i \in [s]$, it holds that

1. $s \leq \alpha$
2. $\forall i \in [s],\ \mathbf{G}_i \in \overline{\mathcal{T}}^{(\beta)}$
3. $\forall i \in [s],\ G_i$ is a subgraph of G
4. $\forall i, j \in [s],\ i \neq j \Rightarrow X_i \cap X_j = \emptyset$
5. $|V(G) \setminus \bigcup_{i \in [s]} X_i| \leq \alpha$
6. $\forall i \in [s],\ \mathbf{tw}(G[X_i]) \leq \gamma$.

We cal the set $V(G) \setminus \bigcup_{i \in [s]} X_i$ *center* of the above (α, β, γ)-protrusion decomposition. Protrusion decompositions have been introduced in [4] in the context of kernelization algorithms (see also [25, 23]). The above definition is a modification of the original one in [4], adapted for the needs of our proofs. The only essential modification is the parameter γ, used in the last requirement. Intuitively, γ bounds the "internal" treewidth of each protrusion $\mathbf{B}_i$.

2.2 Equivalence on boundaried structures.

(Counting) Monadic Second Order Logic. We say that a MSOL-formula ϕ is a *formula on structures* if it has a free variable corresponding to a set of vertices. A structure $\mathbf{G} = (G, A)$ is a *model* for such a formula ϕ, we write this $\mathbf{G} \models \phi$, if it becomes true on G when instantiating the free variable of ϕ by the set A. Given a MSOL-formula ϕ we denote by $|\phi|$ the length of the formula.

Equivalences between b-structures and b-graphs. Let ϕ be a MSOL-formula and $t \in \mathbb{N}$. Given two b-structures $\mathbf{G}_1, \mathbf{G}_2 \in \mathcal{B}^{(t)}$, we say that $\mathbf{G}_1 \equiv_{\phi,t} \mathbf{G}_2$ if

- $\mathbf{G}_1 \sim \mathbf{G}_2$ and
- $\forall \mathbf{F} \in \mathcal{B}^{(t)}\ \mathbf{F} \sim \mathbf{G}_1 \Rightarrow (\mathbf{F} \oplus \mathbf{G}_1 \models \phi \iff \mathbf{F} \oplus \mathbf{G}_2 \models \phi)$

Notice that $\equiv_{\phi,t}$ is an equivalence relation on $\mathcal{B}^{(t)}$. The following result is widely known as Courcelle's theorem and was proven [10]. The same result was essentially proven in [7] and [2]. The version on structures that we present below appeared in [4, Lemma 3.2.].

▶ **Proposition 3.** *There exists a computable function $\xi : \mathbb{N}^2 \to \mathbb{N}$ such that for every CMSO-formula ϕ and every $t \in \mathbb{N}$, the equivalence relation $\equiv_{\phi,t}$ has at most $\xi(|\phi|, t)$ equivalence classes.*

Given a MSOL-formula ϕ and under the light of Proposition 3, we consider a (finite) set $\mathcal{R}_{\phi,t}$ containing one minimum-size member from each of the equivalence classes of $\equiv_{\phi,t}$. Keep in mind that $\mathcal{R}_{\phi,t} \subseteq \mathcal{B}^{(t)}$. Notice that for every $\mathbf{G} \in \mathcal{B}^{(t)}$, there is a b-structure in $\mathcal{R}_{\phi,t}$, we denote it by $\mathsf{rep}_{\phi,t}(\mathbf{G})$, such that $\mathsf{rep}_{\phi,t}(\mathbf{G}) \equiv_{\phi,t} \mathbf{G}$.

3 Approximating protrusion decompositions

The main result of this section is a constant-factor approximation algorithm computing a t-treewidth modulator (Lemma 5). Based on this we also derive a constant-factor approximation algorithm for a protrusion decomposition (Theorem 6). For our proofs we need the following lemma that is a consequence of the results in [34].

▶ **Lemma 4.** *For every h-vertex graph H and every $t \in \mathbb{N}$, there exists a constant c and an algorithm that takes as input an H-topological-minor-free graph G and a t-treewidth modulator $X \subseteq V(G)$ and outputs a $(c|X|, c, t)$-protrusion decomposition along with tree decompositions of its b-graphs of width at most c, in $O_{h+t}(n)$ steps.*

As a consequence of Lemma 4, as long as the input graph G has many vertices (linear in k), there is a vertex set Y whose (internal) treewidth is at most t and contains sufficiently many vertices. The key step of the approximation algorithm, to be shown in the next lemma, is to replace $N[Y]$ with a smaller graph of the same 'type'. Two conditions are to be met during the replacement: first, the minimum-size of a t-treewidth modulator remains the same. Secondly, a t-treewidth modulator of the new graph can be 'lifted' to a t-treewidth modulator of the graph before the replacement without increasing the size.

▶ **Lemma 5.** *For every h-vertex graph H and every t, there is a constant c, depending on h and t, and an algorithm that, given a graph $G \in \mathcal{F}_H$ and $k \in \mathbb{N}$, either outputs an t-treewidth-modulator of G of size at most $c \cdot k$ or reports that no t-treewidth modulator of G exists with size at most k. This algorithm runs in $O_{h+t}(n^2)$ steps.*

Notice that the above lemma, with worst running time, is also a consequence of the recent results in [32]. We insist to the above statement of Lemma 5, as we are interested for a quadratic time approximation algorithm for protrusion decompositions. Indeed, based on Lemma 5 we can prove the following that is the main result of this section.

▶ **Theorem 6.** *Let H be an h-vertex graph and ϕ be a MSOL-formula that is treewidth modulable. Then there is a constant c, depending on h and $|\phi|$, and an algorithm that, given an input (G, k) of $\Pi_{\phi,\mathcal{F}_H}$, either reports no $A \subseteq V(G)$ with $(G, A) \models \phi$ has size at most k or outputs a (ck, c, c)-protrusion decomposition of G along with tree decompositions of its b-graphs, each of width at most c. This algorithm runs in $O_{|\phi|+h}(n^2)$ steps.*

4 The compactor

By Theorem 6, we may assume that a (tk, t, t)-protrusion decomposition $\mathbf{G}_1, \ldots, \mathbf{G}_s$ of G, with $\mathbf{G}_i = (G_i, B_i, -)$, is given for some t. For counting the sets $A \subseteq V(G)$ of size at most k with $(G, A) \models \phi$, we view such a set A as a union of $A_0 \cup A_1 \cup \cdots A_s$, where A_0 is the subset of A residing in the the center of the decomposition, and $A_i = A \cap V(\mathbf{G}_i)$ for each $i \in [s]$. Suppose that $A'_i \subseteq V(\mathbf{G}_i)$ for some $i \in [s]$ satisfies $(G_i, B_i, A_i) \equiv_{\phi,t} (G_i, B_i, A'_i)$ and $|A_i| = |A'_i|$. Then, $(A \setminus A_i) \cup A'_i$ has the same size as $|A|$ and we have $(G, A \setminus A_i \cup A'_i) \models \phi$. In other words, A'_i and A_i are indistinguishable when seen from outside of $\mathbf{G}_i$.

The basic idea of the condenser is to replace all the occurrences of such sets A'_i (include A_i itself) with $O(1)$-bit information; that is, the number of such sets, the size of $|A'_i|$, and the equivalence class containing (G_i, B_i, A'_i). Formally, for the given CMSO-formula ϕ and $t \in \mathbb{N}$, we define the function $\#\mathsf{sol}_{\phi,t}$ so that for each $\mathbf{R} \in \mathcal{R}_{\phi,t}$, $\mathbf{G} := (G, B, -) \in \overline{\mathcal{T}}^{(t)}$, we set

$$\#\mathsf{sol}_{\phi,t}(\mathbf{R}, \mathbf{G}, k) \quad = \quad |\{A \in \binom{V(G)}{k} \mid \mathbf{R} \equiv_{\phi,t} (G, B, A)\}|.$$

This function can be fully computed in linear time on a b-graph of bounded treewidth.

▶ **Lemma 7.** *For every CMSO-formula ϕ and every $t \in \mathbb{N}$, there exists an algorithm that, given a $\mathbf{G} \in \overline{\mathcal{T}}^{(t)}$ and a tree decomposition of $\mathbf{G}$ of width at most t, outputs $\#\mathsf{sol}_{\phi,t}(\mathbf{R}, \mathbf{G}, k')$ for every $(\mathbf{R}, k') \in \mathcal{R}_{\phi,t} \times [0, k]$. This computation takes $O_{|\phi|,t}(nk^2)$ steps.*

The proof of Lemma 7 is based on a dynamic programming procedure. This may follow implicitly from the proofs of Courcelle's theorem (see [11, 9]).

We are now in position to prove Theorem 1.

Proof of Theorem 1. We describe a polynomial size compactor (P, M) for $\Pi_{\phi,\mathcal{F}_H}$. Given an input $(G, k) \in \mathcal{F}_H \times \mathbb{N}$, the condenser P of the compactor runs as a first step the algorithm of Theorem 6. If this algorithm reports that there is no set A of size k with $(G, k) \models \phi$, the the condenser outputs \$, i.e., $\mathsf{A}(G, k) = \$$. Suppose now that the output is a (tk, t, t)-protrusion decomposition $\mathbf{G}_1, \ldots, \mathbf{G}_s$ of G, along with the corresponding tree decompositions, for some constant t that depends only on h and $|\phi|$. Let K be the center of this protrusion decomposition and recall that $|K|, s \leq tk$. We set $G_0 = G[K]$ and let $\mathbf{G}_i = (G_i, B_i, -)$ for each $i \in [s]$. We also define $\mathcal{B} = \{B_i, \mid i \in [s]\}$ where B_i is the boundary of $\mathbf{G}_i$, $i \in [s]$. The next step of the condenser is to apply the algorithm of Lemma 7 and compute $\#\mathsf{sol}_{\phi,t}(\mathbf{R}, \mathbf{G}_i, k')$ for every $(\mathbf{R}, k', i) \in \mathcal{R}_{\phi,t} \times [0, k] \times [s]$, in $O_{|\phi|+h}(nk^2)$ steps. The output of the condenser P is

$$\mathsf{P}(G, k) = (G_0, \mathcal{B}, \{\#\mathsf{sol}_{\phi,t}(\mathbf{R}, \mathbf{G}_i, k') \mid (\mathbf{R}, k', i) \in \mathcal{R}_{\phi,t} \times [0, k] \times [s]\}).$$

Clearly, $\mathsf{P}(G, k)$ can be encoded in $O_{|\phi|+h}(k^2)$ memory positions.

We next describe the extractor M of the compactor. For simplicity, we write $z := \mathsf{P}(G, k)$ and we define $\mathsf{M}(\$) = 0$. We assume that there is a fixed labeling λ of G_0. The extractor M first computes the set $\mathcal{A}$ containing all subsets of K of at most k vertices. Notice that $|\mathcal{A}| = 2^{O_{|\phi|+h}(k)}$. Next, for each $A_0 \in \mathcal{A}$, the algorithm builds the set $\mathcal{M}_{A_0}$ containing all mappings $\mathfrak{m} : [s] \to \mathcal{R}_{\phi,t}$ with the property that, for every $i \in [s]$, $(G_0, B_i, A_0) \sim \mathfrak{m}(i)$. As the boundary of $\mathfrak{m}(i)$ induces an identical labeled graph as B_i does, we denote $\mathfrak{m}(i)$ as $(G_i^{\mathfrak{m}}, B_i, A_i^{\mathfrak{m}})$. Notice that $|\mathcal{M}_{A_0}| = 2^{O_{|\phi|+h}(k)}$, for every $A_0 \in \mathcal{A}$.

Let $A_0 \in \mathcal{A}$ and $\mathfrak{m} \in \mathcal{M}_{A_0}$. For each such pair, the extractor runs a routine that constructs an annotated graph $(D^{\mathfrak{m}}, A^{\mathfrak{m}})$ as follows: first it initializes $\mathbf{D}_0^{\mathfrak{m}} = (D_0, A_0^{\mathfrak{m}})$ with $D_0 = G_0$ and $A_0^{\mathfrak{m}} = A_0$. After constructing $\mathbf{D}_i^{\mathfrak{m}} = (D_i, \bigcup_{j \in [i]} A_j^{\mathfrak{m}})$, the routine

sets $\mathbf{D}^{m}_{i+1} = ((D_i, B_{i+1}, -) \oplus (G^{\mathfrak{m}}_{i+1}, B_{i+1}, -), \bigcup_{j\in[i+1]} A^{\mathfrak{m}}_j)$ iteratively from $i = 0$ up to $s-1$. We set $(D^{\mathfrak{m}}, A^{\mathfrak{m}}) = \mathbf{D}^{\mathfrak{m}}_s$. Notice that the routine runs in $O_{|\phi|+h}(k)$ steps and that $|D^{\mathfrak{m}}| = O_{|\phi|+h}(k)$.

The extractor M is defined as

$$\mathsf{M}(z) = \sum_{A_0\in\mathcal{A}} \sum_{\mathfrak{m}\in\mathcal{M}_{A_0}} [(D^{\mathfrak{m}}, A^{\mathfrak{m}}) \models \phi] \cdot \Big(\sum_{\zeta\in\mathcal{K}_{k-|A_0|}} \prod_{i\in[s]} \#\mathsf{sol}_{\phi,t}(\mathfrak{m}(i), \mathbf{G}_i, \zeta(i) + |B_i \cap A_0|) \Big)$$

where $[\cdot]$ is a function indicating whether a sentence is true (=1) or false (=0), and $\mathcal{K}_{\ell-|A_0|}$ is the set of all vectors $\zeta \in [0,k]^s$ such that $\sum_{i\in[s]} \zeta(i) = \ell - |A_0|$.

Having access to $\{\#\mathsf{sol}_{\phi,t}(\mathbf{R}, \mathbf{G}_i, k') \mid (\mathbf{R}, k', i) \in \mathcal{R}_{\phi,t} \times [0,k] \times [s]\}$, we can compute $\mathsf{M}(z)$ in $2^{O_{|\phi|+h}(k)}$ steps. Therefore, the extractor runs in the claimed running time. It remains to prove that $\mathsf{M}(z)$ equals $|\{A \in \binom{V(G)}{k} \mid (G, A) \models \phi\}|$.

Before proceeding, we present a key claim (for the proof, see the full version of the paper).

▶ **Claim 8.** *Let $\mathbf{H}_i = (H_i, B, A_i)$ for $i = 1, 2$ be two compatible b-structures from $\mathcal{B}^{(t)}$. Let $\mathbf{H}'_2 = (H'_2, B, A'_2)$ be a b-structure equivalent with $\mathbf{H}_2$. Then for every $B' \subseteq V(H_1)$ of size at most t, the two b-structures $\mathbf{D}$ and $\mathbf{D}'$ are equivalent under $\equiv_{\phi,t}$, where*

$$\mathbf{D} = ((H_1, B, -) \oplus (H_2, B, -), B', A_1 \cup A_2) \quad \text{and} \quad \mathbf{D}' = ((H_1, B, -) \oplus (H'_2, B, -), B', A_1 \cup A'_2)$$

Now, consider an arbitrary sequence $A'_1, \ldots, A'_s$ of vertex sets with $A'_i \subseteq V(G_i)$, each of which is counted in $\#\mathsf{sol}_{\phi,t}(\mathfrak{m}(i), \mathbf{G}_i, \zeta(i) + |B_i \cap A_0|)$. Claim 8, $[(G^{\mathfrak{m}}, A^{\mathfrak{m}}) \models \phi] = 1$, and $\mathfrak{m}_i \equiv_{\phi,t} (G_i, B_i, A')$ ensure that $(G, A_0 \cup \bigcup_{i\in[s]} A'_i) \models \phi$. Observe that

$$|A_0 \cup \bigcup_{i\in[s]} A'_i| = |A_0| + \sum_{i\in[s]} |A'_i \setminus B_i| = |A_0| + \sum_{i\in[s]} |A_i| = |A_0| + \sum_{i\in[s]} \zeta(i) = k.$$

That is, each combination of A_0, $\mathfrak{m}$, ζ, and a sequence $A'_1, \ldots, A'_s$ contributing 1 to the sum $\mathsf{M}(z)$, a vertex set A of size precisely k can be uniquely defined and we have $(G, A) \models \phi$. Clearly, distinct combinations lead to distinct such sets. Therefore, $|\{A \in \binom{V(G)}{k} \mid (G, A) \models \phi\}|$ is at least the value of $\mathsf{M}(z)$. This completes the proof. ◀

5 Conclusions

Concerning Theorem 1, we stress that the treewidth-modulability condition can be derived by other meta-algorithmic conditions. Such conditions are minor/contraction bidimensionality and linear separability for graphs excluding a graph/apex graph as a minor [23, 25]. This extends the applicability of our meta-algorithmic result to more problems but in more restricted graph classes. Natural follow-up questions are whether the size of the compactor can be made linear and whether its combinatorial applicability can be extended to more general graph classes.

We envision that the formal definition of a compactor that we give in this paper may encourage the research on data-reduction for counting problems. The apparent open issue is whether other problems (or families of problems) may be amenable to this data-reduction paradigm (in particular, the results in [16, 39, 42, 43] can be interpreted as results on polynomial compactors).

Another interesting question is whether (and to which extent) the fundamental complexity results in [8, 3, 26, 15, 5, 18, 33] on the non-existence of polynomial kernels may have their counterpart for counting problems.

References

1 Karl R. Abrahamson and Michael R. Fellows. Finite Automata, Bounded Treewidth and Well-Quasiordering. In Neil Robertson and Paul D. Seymour, editors, *AMS Summer Workshop on Graph Minors, Graph Structure Theory, Contemporary Mathematics vol. 147*, pages 539–564. American Mathematical Society, 1993.

2 Stefan Arnborg, Jens Lagergren, and Detlef Seese. Easy problems for tree-decomposable graphs. *Journal of Algorithms*, 12:308–340, 1991.

3 Hans L. Bodlaender, Rodney G. Downey, Michael R. Fellows, and Danny Hermelin. On problems without polynomial kernels. *J. Comput. Syst. Sci.*, 75:423–434, December 2009. doi:10.1016/j.jcss.2009.04.001.

4 Hans L. Bodlaender, Fedor V. Fomin, Daniel Lokshtanov, Eelko Penninkx, Saket Saurabh, and Dimitrios M. Thilikos. (Meta) Kernelization. *J. ACM*, 63(5):44:1–44:69, 2016.

5 Hans L. Bodlaender, Bart M. P. Jansen, and Stefan Kratsch. Kernelization Lower Bounds by Cross-Composition. *SIAM J. Discrete Math.*, 28(1):277–305, 2014. doi:10.1137/120880240.

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

7 Richard B. Borie, R. Gary Parker, and Craig A. Tovey. Automatic Generation of Linear-Time Algorithms from Predicate Calculus Descriptions of Problems on Recursively Constructed Graph Families. *Algorithmica*, 7:555–581, 1992.

8 Yijia Chen, Jörg Flum, and Moritz Müller. Lower Bounds for Kernelizations and Other Preprocessing Procedures. *Theory Comput. Syst.*, 48(4):803–839, 2011. doi:10.1007/s00224-010-9270-y.

9 B. Courcelle, J.A. Makowsky, and U. Rotics. On the fixed parameter complexity of graph enumeration problems definable in monadic second-order logic. *Discrete Applied Mathematics*, 108(1):23–52, 2001. Workshop on Graph Theoretic Concepts in Computer Science.

10 Bruno Courcelle. The monadic second-order logic of graphs. I. Recognizable sets of finite graphs. *Information and Computation*, 85(1):12–75, 1990.

11 Bruno Courcelle and M. Mosbah. Monadic second-order evaluations on tree-decomposable graphs. *Theor. Comput. Sci.*, 109:49–82, 1993.

12 Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015.

13 Anuj Dawar, Martin Grohe, and Stephan Kreutzer. Locally Excluding a Minor. In *21st IEEE Symposium on Logic in Computer Science (LICS'07)*, pages 270–279. IEEE, New York, 2007.

14 Babette de Fluiter. *Algorithms for Graphs of Small Treewidth*. PhD thesis, Dept. Computer Science, Utrecht University, 1997.

15 Holger Dell. AND-Compression of NP-Complete Problems: Streamlined Proof and Minor Observations. *Algorithmica*, 75(2):403–423, 2016. doi:10.1007/s00453-015-0110-y.

16 Josep Díaz, Maria Serna, and Dimitrios M. Thilikos. Efficient algorithms for counting parameterized list H-colorings. *J. Comput. System Sci.*, 74(5):919–937, 2008. doi:10.1016/j.jcss.2008.02.004.

17 Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013. doi:10.1007/978-1-4471-5559-1.

18 Andrew Drucker. New Limits to Classical and Quantum Instance Compression. *SIAM J. Comput.*, 44(5):1443–1479, 2015. doi:10.1137/130927115.

19 Zdenek Dvorak, Daniel Kral, and Robin Thomas. Deciding First-Order Properties for Sparse Graphs. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, FOCS '10, pages 133–142. IEEE, Washington, DC, USA, 2010. doi:10.1109/FOCS.2010.20.

20 Zdeněk Dvořák, Daniel Král, and Robin Thomas. Testing First-order Properties for Subclasses of Sparse Graphs. *J. ACM*, 60(5):36:1–36:24, October 2013. doi:10.1145/2499483.

21 Jörg Flum and Martin Grohe. Fixed-parameter tractability, definability, and model-checking. *SIAM J. Comput.*, 31(1):113–145, 2001.

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

23 F. V. Fomin, D. Lokshtanov, S. Saurabh, and D. M. Thilikos. Bidimensionality and Kernels. In *21st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2010)*, pages 503–510. ACM-SIAM, 2010.

24 Fedor V. Fomin, Daniel Lokshtanov, Neeldhara Misra, and Saket Saurabh. Planar F-Deletion: Approximation, Kernelization and Optimal FPT Algorithms. In *53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS 2012, New Brunswick, NJ, USA, October 20-23, 2012*, pages 470–479, 2012.

25 Fedor V. Fomin, Daniel Lokshtanov, Saket Saurabh, and Dimitrios M. Thilikos. Bidimensionality and Kernels. *CoRR*, abs/1606.05689, 2016. Revised version. arXiv:1606.05689.

26 Lance Fortnow and Rahul Santhanam. Infeasibility of instance compression and succinct PCPs for NP. *J. Comput. Syst. Sci.*, 77(1):91–106, 2011. doi:10.1016/j.jcss.2010.06.007.

27 Markus Frick. Generalized Model-Checking over Locally Tree-Decomposable Classes. *Theory of Computing Systems*, 37(1):157–191, January 2004.

28 Markus Frick and Martin Grohe. Deciding First-Order Properties of Locally Tree-Decomposable Graphs. In Jirí Wiedermann, Peter van Emde Boas, and Mogens Nielsen, editors, *Automata, Languages and Programming*, volume 1644 of *LNCS*, pages 72–72. Springer Berlin / Heidelberg, 1999.

29 Martin Grohe. Logic, graphs, and algorithms. In *Logic and Automata*, pages 357–422. Amsterdam University Press, 2008.

30 Martin Grohe and Stephan Kreutzer. *Methods for algorithmic meta theorems*, chapter Model Theoretic Methods in Finite Combinatorics, pages 181–206. Contemporary Mathematics, 2011.

31 Martin Grohe, Stephan Kreutzer, and Sebastian Siebertz. Deciding First-order Properties of Nowhere Dense Graphs. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*, STOC '14, pages 89–98, New York, NY, USA, 2014. ACM.

32 Anupam Gupta, Euiwoong Lee, Jason Li, Pasin Manurangsi, and Michal Wlodarczyk. Losing treewidth by separating subsets. *CoRR*, abs/1804.01366, 2018. arXiv:1804.01366.

33 Danny Harnik and Moni Naor. On the Compressibility of *NP* Instances and Cryptographic Applications. *SIAM J. Comput.*, 39(5):1667–1713, 2010. doi:10.1137/060668092.

34 Eun Jung Kim, Alexander Langer, Christophe Paul, Felix Reidl, Peter Rossmanith, Ignasi Sau, and Somnath Sikdar. Linear Kernels and Single-Exponential Algorithms Via Protrusion Decompositions. *ACM Trans. Algorithms*, 12(2):21:1–21:41, 2016.

35 Eun Jung Kim, Maria Serna, and Dimitrios M. Thilikos. Data-compression for parametrized counting problems on sparse graphs. *CoRR*, abs/1809.08160, 2018. arXiv:1809.08160.

36 Stephan Kreutzer. Algorithmic Meta-theorems. In *IWPEC*, pages 10–12. Springer, 2008.

37 Daniel Lokshtanov, Neeldhara Misra, and Saket Saurabh. Kernelization - Preprocessing with a Guarantee. In Hans L. Bodlaender, Rod Downey, Fedor V. Fomin, and Dániel Marx, editors, *The Multivariate Algorithmic Revolution and Beyond*, pages 129–161. Springer-Verlag, Berlin, Heidelberg, 2012. URL: http://dl.acm.org/citation.cfm?id=2344236.2344248.

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

39 Naomi Nishimura, Prabhakar Ragde, and Dimitrios M. Thilikos. Parameterized Counting Algorithms for General Graph Covering Problems. In Frank K. H. A. Dehne, Alejandro López-Ortiz, and Jörg-Rüdiger Sack, editors, *Algorithms and Data Structures, 9th International Workshop, WADS 2005, Waterloo, Canada, August 15-17, 2005, Proceedings*, volume 3608 of *Lecture Notes in Computer Science*, pages 99–109. Springer, 2005. doi:10.1007/11534273_10.

40 D. Seese. The structure of the models of decidable monadic theories of graphs. *Annals of Pure and Applied Logic*, 53(2):169–195, 1991. doi:10.1016/0168-0072(91)90054-P.

41 Detlef Seese. Linear Time Computable Problems and First-Order Descriptions. *Mathematical Structures in Computer Science*, 6(6):505–526, 1996.

42 Marc Thurley. Kernelizations for parameterized counting problems. In *4th international conference on Theory and applications of models of computation*, TAMC'07, pages 705–714. Springer-Verlag, Berlin, Heidelberg, 2007. URL: http://portal.acm.org/citation.cfm?id=1767854.1767920.

43 Marc Thurley. Tractability and Intractability of Parameterized Counting Problems, May 2006. Diploma thesis.