



HAL
open science

Compactors for parameterized counting problems

Dimitrios M. Thilikos

► **To cite this version:**

Dimitrios M. Thilikos. Compactors for parameterized counting problems. *Computer Science Review*, 2021, 39, pp.100344. 10.1016/j.cosrev.2020.100344 . hal-03093911

HAL Id: hal-03093911

<https://hal.science/hal-03093911v1>

Submitted on 22 Oct 2021

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Compactors for Parameterized Counting Problems


Dimitrios M. Thilikos*

LIRMM, Univ Montpellier, CNRS, Montpellier, France.

Abstract

The concept of *compactor* has been introduced in [Eun Jung Kim, Maria J. Serna, and Dimitrios M. Thilikos. *Data-compression for parametrized counting problems on sparse graphs. ISAAC 2018, LIPIcs Vol. 123:20:1–20:13*] as a general data-reduction concept for parametrized counting problems. For a function $F : \Sigma^* \rightarrow \mathbb{N}$ and a parameterization $\kappa : \Sigma^* \rightarrow \mathbb{N}$, a compactor (C, E) consists of a polynomial-time computable function P , called *condenser*, and a computable function M , called *extractor*, such that $F = M \circ P$. If the size of $P(x)$ is bounded by a polynomial function of $\kappa(x)$, then we say that the compactor (C, E) is of polynomial size. Compactors can be seen as an attempt to formalize the notion of preprocessing for counting problems.

Keywords: Parameterized algorithms, counting algorithms, compactor, Graph Algorithms

arameterized complexity was introduced as a multi-variable framework for dealing with the inherent complexity of computational problems. It was invented by Downey and Fellows in their pioneer work in [34, 35, 33, 1] and currently constitutes a fully developed discipline of Theoretical Computer Science (see [22, 41, 59, 36] for related textbooks). Parameterized complexity proposed a refined analysis of computational problems. The idea is to study computational problems as “bivariate entities”: apart from the input size, say n , we consider as a second variable a *parameter* k that measures certain characteristics of the input. The general question is whether a problem parameterization is *fixed parameter tractable*, i.e., it can

*Supported by projects DEMOGRAPH (ANR-16-CE40-0028) and ESIGMA (ANR-17-CE23-0010) and by the Research Council of Norway and the French Ministry of Europe and Foreign Affairs, via the Franco-Norwegian project PHC AURORA 2019.

Email address: sedthilk@thilikos.info (Dimitrios M. Thilikos)

be solved by an algorithm running in time $f(k) \cdot n^{O(1)}$ or, in other words, it admits a uniformly polynomial algorithm for every fixed value of the parameter k . This *multi-variate* complexity theory provides a more precise view on problem complexity and offers additional tools to deal with their hardness.

Parameterized complexity also offered a theoretical base for the formalization of the notion of preprocessing. A well-studied concept of data-reduction in parameterized complexity is *kernelization*. A *kernelization algorithm* is a polynomial-time algorithmic reduction of a problem to itself so that the new instances have size that depends *exclusively* on the parameter. When this function is polynomial, then we have a *polynomial kernel*. A polynomial kernelization permits a significant data-reduction of the problem instances. 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. Kernelization has been extensively studied in parametrized complexity and offered polynomial kernels for a large variety of problems (see [46] for a recent textbook on this subject).

A, relatively small, part of the research on parameterized computation has been focused to the study of parameterized counting problems [40, 56, 4, 16, 19, 18, 17, 15, 14, 21, 20, 6, 9, 57, 52, 51]. Moreover, even less effort has been done for the definition and study of data-reduction concepts for parameterized counting problems. The purpose of this survey is to present the notion of a *compactor*, formally introduced in [54], as an algorithmic paradigm of data-reduction for parameterized counting problems [66, 67, 27, 29, 60].

Our presentation is intended to be self-contained. We provide in Section 1 the formal definitions of a parameterized (counting) problem and the notions of FPT-algorithm and kernelization algorithm. We also provide a series of working examples of parameterized (counting) problems on graphs. In Section 2 we give the formal definition of a compactor and present earlier results on data-reduction schemes for parameterized counting problems. Finally, Section 3 contains a general algorithmic meta-theorem on the automated derivation of a compactor that has recently appeared in [54] and is based on earlier meta-algorithmic results in [7, 45, 44, 53].

1. General context

This section contains the definition of all combinatorial and algorithmic concepts that are necessary in order to present the results of this paper. We use notation \mathbb{N} in order to denote the set of non-negative integers. Also, we set $\mathbb{N}^+ = \mathbb{N} \setminus \{0\}$ and we denote by **poly** the set of all polynomials.

1.1. Some definitions on graphs.

Many interesting computational problems can be stated with the help of graphs. All graphs in this paper are undirected, simple, and finite. Given a graph $G = (V, E)$, we denote $V(G) = V$ and $E(G) = E$. Given some $S \subseteq V(G)$, we denote by $G \setminus S$ the graph obtained if we remove from G the vertices in S , along with their incident edges. We also denote $G[S] = G \setminus (V(G) \setminus S)$ and we call $G[S]$ the *subgraph of G induced by S* . If G' is a graph where $V(G') \subseteq V(G)$ and $E(G') \subseteq E(G[V(G')])$ then we say that G' is a *subgraph of G* . We define the *closed neighborhood* of S in G , denoted by $N_G[S]$, as the set of all endpoints of edges containing some vertex in S as endpoint. The *open neighborhood* of S in G is defined as $N_G(S) := N_G[S] \setminus S$. Given a vertex $v \in V(G)$ we set $N_G(v) := N_G(\{v\})$. We call $N_G(v)$ the *neighborhood of v in G* and we call the vertices of $N_G(v)$ the *neighbors of v in G* . We define the *degree* of a vertex v in a graph G by $\deg_G(v) := |N_G(v)|$ and we set $\Delta(G) = \max\{\deg_G(v) \mid v \in V(G)\}$. Given a $d \in \mathbb{N}$, we recursively define $N_G^d[S] := N_G[N_G^{d-1}(S)]$, starting from $N_G^{(0)}(S) := S$ (notice that $N_G^{(1)}(S) = N_G(S)$). We denote $\binom{V(G)}{k} = \{S \mid S \subseteq V(G) \text{ and } |S| = k\}$ and we use \mathcal{G} for the set of all graphs.

A graph G on n vertices is *connected* if for every $v \in V(G)$, $N_G^{n-1}(\{v\}) = V(G)$. A *cycle of G* is any connected subgraph of G with all its vertices of degree 2. We say that a graph is *acyclic* if it does not contain any cycle as a subgraph. A path P in G is any acyclic connected subgraph with at most two vertices of degree 1; such vertices are called *the endpoints of G* . We use $K_r, r \in \mathbb{N}^+$ to denote the complete graph on r vertices.

We consider two general families of graph classes based on the minor and the topological minor relations. Given two graphs G_1 and G_2 , we say that G_1 is a *minor* of G_2 if G_1 can occur from some subgraph of G_2 after contracting edges. Also we say that G_1 is a *topological minor* of G_2 if G_2 contains as a subgraph some subdivision¹ of G_1 . Given a finite set of graphs \mathcal{H} , we denote

¹A *subdivision* of a graph H is any graph obtained from H after replacing edges with

by $\mathcal{M}_{\mathcal{H}}$ the set of all graphs excluding all the graphs in \mathcal{H} as a minor and by $\mathcal{T}_{\mathcal{H}}$ the set of all graphs excluding all the graphs in \mathcal{H} as a topological minor. We use \mathcal{P} for the class of planar graphs, i.e., $\mathcal{P} := \mathcal{T}_{\{K_{3,3}, K_5\}}$ (that is also equal to $\mathcal{M}_{\{K_{3,3}, K_5\}}$). Given a graph G and a set $S \subseteq V(G)$, we say that S is a *planarizer of G* if $G \setminus S \in \mathcal{P}$.

1.2. Parameterized problems

We see a decision problem as a language $L \subseteq \Sigma^*$. Here, Σ is an alphabet that is used to encode the instances of the problem L . A *problem parameterization* is a function $\kappa : \Sigma^* \rightarrow \mathbb{N}$, i.e., a mapping assigning a non-negative integer to each problem instance. A *parameterized problem* is a pair $\Pi = (L, \kappa)$ where L is a problem and κ is a problem parameterization.

Parameterized problems on Graphs.. An *annotated graph* is a pair (G, A) where G is a graph and $A \subseteq V(G)$. Given a set \mathcal{A} of annotated graphs and a class of graphs $\mathcal{Z} \subseteq \mathcal{G}$, we define the restriction of \mathcal{A} to \mathcal{Z} by $\mathcal{A} \upharpoonright \mathcal{Z} = \{(G, S) \mid (G, S) \in \mathcal{A} \wedge G \in \mathcal{Z}\}$. We define the function $F_{\mathcal{A}} : \mathcal{G} \times \mathbb{N} \rightarrow \mathbb{N}$ so that

$$F_{\mathcal{A}}(G, k) = |\{S \mid (G, S) \in \mathcal{A} \text{ and } |S| = k\}|. \quad (1)$$

In other words, $F_{\mathcal{A}}(G, k)$ counts the members of $\binom{V(G)}{k}$ that together with G form an annotated graph in \mathcal{A} . We consider the following general parameterized problem.

$\Pi_{\mathcal{A}}$
Instance: a graph G and a $k \in \mathbb{N}$.
Parameter: k .
Question: $F_{\mathcal{A}}(G, k) > 0$?

The above parameterized problem can be seen as the pair $\Pi_{\mathcal{A}} = (L_{\mathcal{A}}, \kappa)$ where

$$L_{\mathcal{A}} = \{\langle G, k \rangle \mid F_{\mathcal{A}}(G, k) > 0\} \text{ and } \kappa(\langle G, k \rangle) = k.$$

Recall that $\langle G, k \rangle$ encodes both the graph G and the integer k in the alphabet Σ . From now on, whenever we refer to a parameterized problem on graphs

paths on the same endpoints.

we use n for the number of vertices of the input graph and we measure the complexity of an algorithm for this problem as a function of n and the parameter k . We can define the restriction of a parameterized problem $\Pi_{\mathcal{A}}$ to some graph class \mathcal{Z} by setting $\Pi_{\mathcal{A}} \upharpoonright \mathcal{Z} := \Pi_{\mathcal{A} \upharpoonright \mathcal{Z}}$

The definition of $\Pi_{\mathcal{A}}$ encompasses several parameterized problems on graphs. We give a series of examples of such problems.

- p -VERTEX COVER: $\Pi_{\text{vc}} := \Pi_{\mathcal{A}_{\text{vc}}}$ where $\mathcal{A}_{\text{vc}} = \{(G, S) \mid E(G \setminus S) = \emptyset\}$.
- p -FEEDBACK VERTEX SET: $\Pi_{\text{fvs}} := \Pi_{\mathcal{A}_{\text{fvs}}}$ where $\mathcal{A}_{\text{fvs}} = \{(G, S) \mid G \setminus S \text{ is acyclic}\}$.
- p -DOMINATING SET: $\Pi_{\text{ds}} := \Pi_{\mathcal{A}_{\text{ds}}}$ where $\mathcal{A}_{\text{ds}} = \{(G, S) \mid V(G) = N_G[S]\}$.
- p -CYCLE DOMINATION, is $\Pi_{\text{cd}} := \Pi_{\mathcal{A}_{\text{cd}}}$ where $\mathcal{A}_{\text{cd}} = \{(G, S) \mid G \setminus N_G[S] \text{ is acyclic}\}$.
- p -INDEPENDENT SET: $\Pi_{\text{is}} := \Pi_{\mathcal{A}_{\text{is}}}$ where $\mathcal{A}_{\text{is}} = \{(G, S) \mid E(G[S]) = \emptyset\}$.
- p -LONGEST CYCLE, is $\Pi_{\text{lc}} := \Pi_{\mathcal{A}_{\text{lc}}}$ where $\mathcal{A}_{\text{lc}} = \{(G, S) \mid G[S] \text{ has a cycle on } |S| \text{ vertices}\}$.

In the notation of the above problems, the prefix “ p –” is used to denote that we consider a parameterized problem with the standard parameterization $\kappa(\langle G, k \rangle) = k$.

Problem variants.. Recall that, in the definition of F in (1), we count sets of size *exactly* k . Alternatively, we define $F_{\mathcal{A}}^{(\leq)}$ or $F_{\mathcal{A}}^{(\geq)}$ by considering in (1) sets of size $\geq k$ or $\leq k$, respectively. That way we may define different parameterized problems, denoted by $\Pi_{\mathcal{A}}^{(\leq)}$ and $\Pi_{\mathcal{A}}^{(\geq)}$.

Notice that $\Pi_{\bullet}^{(\leq)} = \Pi_{\bullet}$ when $\bullet \in \{\text{vc}, \text{vfs}, \text{ds}, \text{cd}\}$. The reason for this is that \mathcal{A}_{\bullet} is *anti-monotone*: if $(G, S) \in \mathcal{A}_{\bullet}$ and $S' \supseteq S$, then $(G, S') \in \mathcal{A}_{\bullet}$. On the other side \mathcal{A}_{is} is *monotone*: if $(G, S) \in \mathcal{A}_{\text{is}}$ and $S' \subseteq S$, then $(G, S') \in \mathcal{A}_{\text{is}}$ and this implies that $\Pi_{\text{is}}^{(\leq)}$ is a trivial problem (every instance is a **yes**-instance). Notice that \mathcal{A}_{lc} is neither monotone or anti-monotone. $\Pi_{\text{lc}}^{(\leq)}$ is different than Π_{lc} as **yes**-instances of Π_{lc} are also a **yes**-instances of $\Pi_{\text{lc}}^{(\leq)}$ but not vice-versa.

Notice now that, because of the anti-monotonicity of \mathcal{A}_\bullet when $\bullet \in \{\text{vc}, \text{vfs}, \text{ds}, \text{cd}\}$, $\Pi_\bullet^{(\geq)}$ is trivial (the answer is always yes). On the other side, $\Pi_{\text{is}} = \Pi_{\text{is}}^{(\geq)}$ because of the monotonicity of \mathcal{A}_{is} . Finally, observe that $\Pi_{\text{lc}}^{(\geq)}$ is different from both $\Pi_{\text{lc}}^{(\leq)}$ and Π_{lc} .

Parameterized algorithms. Let $\Pi = (L, \kappa)$ be a parameterized problem. We say that Π is *fixed parameter tractable* if there exists a function $f : \mathbb{N} \rightarrow \mathbb{N}$ and an algorithm (called *FPT-algorithm*) that, with input $x \in \Sigma^*$, outputs whether $x \in L$ in $f(\kappa(x)) \cdot n^{O(1)}$ steps. The parameterized complexity class **FPT** is defined as the class of all parameterized problems that are fixed parameter tractable. The central question in Parameterized Complexity is to classify parameterized problems in the class **FPT** or to provide evidence (or proof) that this is not possible. For example, it is known that $\Pi_{\text{vc}} \in \text{FPT}$ and the best, so far, **FPT**-algorithm for proving this, runs in $O(1.2738^k + k \cdot n)$ steps [12]. On the negative side, there is a hierarchy of parameterized complexity classes, known as *the W-hierarchy*, as well as appropriate parameterized reductions, that permit the proof of hardness results. For example, it is known that Π_{ds} , Π_{cd} , and Π_{is} are $W[2]$ -complete, implying that the existence of an **FPT**-algorithm for p -DOMINATING SET or p -CYCLE DOMINATION or p -INDEPENDENT SET is highly unexpected. On the other side, if we restrict these three problems to the class of planar graphs then they become fixed parameter tractable, i.e., $\Pi_{\text{ds}} \cap \mathcal{P} \in \text{FPT}$, $\Pi_{\text{cd}} \cap \mathcal{P} \in \text{FPT}$, and $\Pi_{\text{is}} \cap \mathcal{P} \in \text{FPT}$. Finally, Π_{vfs} and Π_{lc} belong both to **FPT**.

Kernelization. Let $\Pi = (L, \kappa)$ be a parameterized problem. A *kernelization algorithm* (or simply a *kernelization*) for Π is a polynomial-time computable function $A : \Sigma^* \rightarrow \Sigma^*$ such that

- $\forall x \in \Sigma^* \quad x \in L \iff A(x) \in L$ (i.e., x and $A(x)$ are equivalent instances of L) and
- there exists a computable function $s : \mathbb{N} \rightarrow \mathbb{N}$ such that $\forall x \in \Sigma^* \quad |A(x)| \leq s(\kappa(x))$.

We call the function s *the size* of the kernelization A . If $s \in \text{poly}$ we say that A is a *polynomial-size kernelization* for Π .

It is known that every parameterized problem that is in **FPT** admits a kernelization and vice versa. However, it is not always the case that there is a kernelization of polynomial size. A central question of parameterized

computation is to distinguish which problems in FPT have kernelizations of polynomial size and which do not. For example, Π_{vc} , $\Pi_{\text{ds}} \cap \mathcal{P}$, $\Pi_{\text{cd}} \cap \mathcal{P}$, and $\Pi_{\text{is}} \cap \mathcal{P}$ have kernelizations whose size is a linear function of $k = \kappa(x)$ (see e.g., [58, 11, 46]), while Π_{vs} has a kernelization of quadratic size due to [65]. On the negative side, the problem Π_{lc} , while being in FPT (actually even the problem $\Pi_{\text{lc}} \cap \mathcal{P}$) is not expected to have a kernelization of polynomial size [46].

1.3. Parameterized counting problems

A *counting problem* is a function $F : \Sigma^* \rightarrow \mathbb{N}$ for which there is a binary relation $R \subseteq \Sigma^* \times \Sigma^*$ such that $\forall x \in \Sigma^*$, $F(x) = |\{y \mid (x, y) \in R\}|$.

Following the parameterization idea, we define a *parameterized counting problem* as a pair (F, κ) where $F : \Sigma^* \rightarrow \mathbb{N}$ is a counting problem and $\kappa : \Sigma^* \rightarrow \mathbb{N}$ is a problem parameterization.

If, in the definition of $\Pi_{\mathcal{A}}$, we ask as an answer the value of $F_{\mathcal{A}}(G, k)$, then we define a parameterized counting problem that we denote by $\#\Pi_{\mathcal{A}}$. Formally, $\#\Pi_{\mathcal{A}} = (F_{\mathcal{A}}, \kappa)$ where $\kappa(\langle G, k \rangle) = k$. To see $\#\Pi_{\mathcal{A}}$ as a counting problem, we consider the relation $R_{\mathcal{A}} \subseteq \Sigma^* \times \Sigma^*$ where $(x, y) \in R_{\mathcal{A}}$ if and only if $x = \langle G, k \rangle$ and $y = \langle S \rangle$, where $S \in \binom{V(G)}{k}$ and $(G, S) \in \mathcal{A}$. Clearly, $F_{\mathcal{A}}(G, k) = |\{y \mid (x, y) \in R_{\mathcal{A}}\}|$.

Given a parameterized counting problem $\#\Pi = (F, \kappa)$, we say that $\#\Pi \in \#\text{FPT}$ if the function F is computable in $f(\kappa(x)) \cdot n^{O(1)}$ steps. At this point we have to stress that, when evaluating the running time, we adopt the standard Uniform Cost Measure (UCM) model where all basic arithmetic computations are carried out in constant time and we assume that numerical values that are produced during the execution of the algorithm can be stored in constant space [2].

We may also define the variants $\#\Pi_{\mathcal{A}}^{(\leq)} = (F_{\mathcal{A}}^{(\leq)}, \kappa)$ and $\#\Pi_{\mathcal{A}}^{(\geq)} = (F_{\mathcal{A}}^{(\geq)}, \kappa)$. Notice that if \mathcal{A} is anti-monotone, then $\#\Pi_{\mathcal{A}}^{(\leq)} \in \text{FPT} \iff \#\Pi_{\mathcal{A}} \in \text{FPT}$. To see this, notice that $F_{\mathcal{A}}^{(\leq k)}(G, k) = \sum_{i \in \{0, \dots, k\}} F_{\mathcal{A}}(G, i)$, while $F_{\mathcal{A}}(G, k) = F_{\mathcal{A}}^{(\leq)}(G, k) - F_{\mathcal{A}}^{(\leq)}(G, k - 1)$, when $k \geq 1$. Interestingly, this situation is not symmetric for $\#\Pi_{\mathcal{A}}^{(\geq)}$ when \mathcal{A} is monotone. The fact that $F_{\mathcal{A}}(G, k) = F_{\mathcal{A}}^{(\geq)}(G, k) - F_{\mathcal{A}}^{(\geq)}(G, k + 1)$ implies that $\#\Pi_{\mathcal{A}}^{(\geq)} \in \text{FPT} \Rightarrow \#\Pi_{\mathcal{A}} \in \text{FPT}$ however the fact that $F_{\mathcal{A}}^{(\geq)}(G, k) = \sum_{i \in \{k, \dots, n\}} F_{\mathcal{A}}(G, i)$ does not imply that $\#\Pi_{\mathcal{A}} \in \text{FPT} \Rightarrow \#\Pi_{\mathcal{A}}^{(\geq)} \in \text{FPT}$.

1.4. An FPT-algorithm for $\#p$ -VERTEX COVER

We next give an example of an easy algorithm for the $\#p$ -VERTEX COVER problem (i.e. the problem $\#\Pi_{\text{vc}}$).

Suppose that (G, k) is an instance of $\#\Pi_{\text{vc}}$. A *high-degree vertex* for (G, k) is a vertex v of G with more than k neighbors. Observe that if v is a high-degree vertex for (G, k) , then v should be a member of every vertex cover of G of size k , therefore $F_{\text{Avc}}(G, k) = F_{\text{Avc}}(G \setminus \{v\}, k - 1)$. By repetitively applying this reduction, we are able to reduce, in polynomial-time, the initial instance (G, k) of the problem to a new one, say (G', k') , where $F_{\text{Avc}}(G, k) = F_{\text{Avc}}(G', k')$ and such that $\Delta(G') \leq k' \leq k$. Let now I be the set of isolated vertices of G' . Let G'' be the graph obtained if we discard from G' $\min\{0, |I| - k\}$ vertices from I . We denote by z the number of discarded vertices. Because no more than k vertices from I can participate to a vertex cover of G' of size k , it follows that (G'', k') and (G, k) are equivalent instances of Π_{vc} . Moreover, it can be easily proved that $|V(G'')| \leq k^2 + 2 \cdot k$ and $|E(G'')| \leq k^2$, therefore the above reduction procedure can be seen as a polynomial size kernelization for the decision problem Π_{vc} . This kernelization is known as Buss' kernelization [10] and has extensively been studied (and extended) in the context of kernelization algorithms.

Recall that our purpose is to design an algorithm for the counting problem $\#\Pi_{\text{vc}}$. For this, is enough to observe that that

$$F_{\text{Avc}}(G', k') = \sum_{i \in \{0, \dots, k\}} F_{\text{Avc}}(G'', i) \cdot \binom{z}{k' - i} \quad (2)$$

This means that the right part of (2) can be computed by enumerating all subsets of $V(G'')$ of size $i \in \{0, \dots, k\}$. As $|V(G'')| \leq k^2 + 2 \cdot k$, we need to consider, for each $i \in \{1, \dots, k\}$, at most $\binom{k^2 + 2 \cdot k}{i}$ subsets of $V(G'')$. Therefore, given G'', k' , and z , we can compute $F_{\text{Avc}}(G', k')$, and therefore $F_{\text{Avc}}(G, k)$ as well, in $2^{O(k^2)}$ steps. This implies that $F_{\text{Avc}}(G, k)$ can be computed in $2^{O(k^2)} + |V(G)|^{O(1)}$ steps. Therefore, $\#\Pi_{\text{vc}} \in \#\text{FPT}$.

The above algorithm can be easily adapted for $\#\Pi_{\text{vc}}^{(\leq k)}$ and also for $\#\Pi_{\text{vc}}^{(\geq k)}$ by using inclusion-exclusion arguments.

2. Compactors for counting problems

In this section we give the definition of the concept of *compactor* as this recently appeared in [54]. We also fit in this framework previous results on

data-reduction for parameterized counting problems.

2.1. Compactors

Notice that the FPT-algorithm for $\#p$ -VERTEX COVER in Subsection 1.4 has a flavor of preprocessing similar to that of the kernelization for its decision counterpart. The preprocessing step computes, in polynomial-time, the graph G'' and the number z and then, based on G'' and z , the value of $F_{\text{Avc}}(G, k)$ can be computed in $2^{O(k^2)}$ steps. Our intention is to fit this in a more general framework. For this, we give the formal definitions of a compactor as a possible formalization of the notion of preprocessing and data-reduction for counting problems.

Let (F, κ) be a parameterized counting problem. A *compactor* for (F, κ) is a pair (\mathbf{C}, \mathbf{E}) where

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

We call the function s *size* of the compactor (\mathbf{C}, \mathbf{E}) and, if $s \in \text{poly}$, we say that (\mathbf{C}, \mathbf{E}) is a *polynomial-size compactor* for (F, κ) . We call the running time of the algorithm computing \mathbf{C} , measured as a function of $|x|$, *condensing time* of (\mathbf{C}, \mathbf{E}) . We also call the running time of the algorithm computing \mathbf{E} , measured as a function of the parameter $\kappa(x)$, *extracting time* of (\mathbf{C}, \mathbf{E}) .

Recall that for parameterized decision problems, FPT-membership is equivalent to the existence of a kernelization. Interestingly, the same holds for parameterized counting problems when we replace kernelization by compactor.

Lemma 1 ([54]). *A parameterized counting problem has an FPT-algorithm if and only if there is a compactor for it.*

Proof. Let (F, κ) be a parameterized counting problem. Suppose that an algorithm A computes $F(x)$ in time $f(\kappa(x))|x|^{O(1)}$ for every input $x \in \Sigma^*$. We define the condenser \mathbf{C} as the function

$$\mathbf{C}(x) = \begin{cases} F(x) & \text{if } |x| > f(\kappa(x)) \\ x & \text{otherwise.} \end{cases}$$

Clearly, $\mathbf{C}(x)$ can be computed in polynomial-time since, if $|x| > f(\kappa(x))$, then one can compute $F(x)$ using algorithm A in time $f(\kappa(x))|x|^{O(1)} = |x|^{O(1)}$. Furthermore, $\mathbf{C}(x)$ has length at most $f(\kappa(x))$. For the extractor \mathbf{E} , we define the image of $z = \mathbf{C}(x)$ as

$$\mathbf{E}(z) = \begin{cases} z & \text{if } z \text{ is a numerical value} \\ F(z) & \text{otherwise.} \end{cases}$$

Note that the function \mathbf{E} can be computed; in particular $F(z)$ can be computed by algorithm A . Clearly, we have $F = \mathbf{E} \circ \mathbf{C}$ and (\mathbf{C}, \mathbf{E}) is a compactor for (F, κ) .

Conversely, let (\mathbf{C}, \mathbf{E}) be a compactor for (F, κ) and let the function s be the size of the compactor. For every input $x \in \Sigma^*$, we can run an algorithm in time $O(|x|^{O(1)})$ to compute $\mathbf{C}(x)$ and an algorithm in time $g(|\mathbf{C}(x)|)$ to compute $\mathbf{E}(\mathbf{C}(x)) = F(x)$. As $|\mathbf{C}(x)| \leq s(\kappa(x))$ and the function g can be assumed to be non-decreasing, this computes $F(x)$ in time $O(|x|^{O(1)} + (g \circ s \circ \kappa)(x))$. \square

Let's try now to fit the previous algorithm for $\#p$ -VERTEX COVER problem to the setting of a compactor. The condenser $\mathbf{C} : \Sigma^* \rightarrow \Sigma^*$ receives as input the instance $x = \langle G, k \rangle$ and outputs the triple $\mathbf{C}(x) = \langle G', k', s \rangle$. Also the extractor is the algorithm that, given $z = \langle G', k', s \rangle$ evaluates (2) in $2^{O(k^2)}$ steps. We have that $F_{\mathcal{A}_{vc}}(x) = \mathbf{E}(z) = (\mathbf{E} \circ \mathbf{C})(x)$. As $|G'| = O(k^2)$ and the numbers k' and s are stored in constant space, we have that $|\mathbf{C}(x)| = O(k^2) = O((\kappa(x))^2)$, therefore (\mathbf{C}, \mathbf{E}) is a compactor of quadratic size.

2.2. Some (pre)history

We now give some examples of parameterized counting problems where FPT-algorithms have been obtained by making use of data-reduction techniques that can be interpreted as compactor algorithms.

Counting list H -colorings.. The first trace of a data-reduction algorithm for counting parameterized problems appeared in [27, 29]. Let (H, C) be a pair where H is a fixed graph that may have loops, $C \subseteq V(H)$, and consider the following counting parameterized problem.

#LIST (H, C) -COLORING

Instance: a graph G , a function $L : V(G) \rightarrow 2^{V(H)}$, and a function $K : C \rightarrow \mathbb{N}$.

Parameter: $k := \sum_{a \in C} K(a)$.

Question: the number of different functions $\chi : V(G) \rightarrow V(H)$ such that

1. for every edge $\{v, u\}$ of G , $\{\chi(v), \chi(u)\}$ is an edge of H .
2. for every $v \in V(G)$, $\chi(v) \in L(v)$
3. for every $a \in C$, $|\chi^{-1}(a)| = K(a)$.

The function L assigns to each vertex of G a list of vertices of H . Also the function K can be seen as a partial weighting of H on the vertices of C . Condition (1) demands that χ is an H -coloring of G , i.e., a coloring of G by the vertices of H so that every edge of G is mapped to an edge of H . Condition (2) demands that each H -coloring χ is a list H -coloring where the color assigned to the vertices of G respect the list L . Lastly, Condition (3), demands that the number of vertices of G that are colored by a vertex a of C is equal to the weight, according to K , of vertex a . LIST (H, C) -COLORING can be seen as a general parameterization of the LIST H -COLORING problem, whose complexity has been determined in [28] (see also [31, 32, 30]). If one insists to express the LIST (H, C) -COLORING using the general formalism that we adapted for counting parameterized problems, we can see it as the pair $\#\Pi_{(H,C)} = (F_{(H,C)}, \kappa_{(H,C)})$ where

$$F_{(H,C)}(\langle G, L, K \rangle) = |\{\chi : V(G) \rightarrow V(H) \mid (1)-(3) \text{ hold}\}| \text{ and}$$

$$\kappa_{(H,C)}(\langle G, L, K \rangle) = \sum_{a \in C} K(a).$$

In [27, 29] it is proved that if $E(H \setminus C) \neq \emptyset$ or H is a reflexive clique or H is a complete bipartite graph, then $\#\Pi_{(H,C)} \in \#\text{FPT}$. The proof strategy in [27, 29] was to construct a new graph \tilde{G} , called *tribal graph*, on $2^{O(k+|H|)}$ vertices and reduce the problem of counting list (H, C) -colorings on G to the problem of enumerating list (H, C) -colorings on \tilde{G} . The authors of [27, 29] called this technique *compactor enumeration*, to stress the fact that the tribal graph \tilde{G} contains a certificate for each class in a suitable partition of the solution space of the initial problem. The graph \tilde{G} plus some adequate numeric information storing the correspondence between solutions on \tilde{G} (specified in [27, 29]) and classes of solutions in G can be seen as a compactor of size $2^{O(k+|H|)}$.

Another example of a parameterized counting problem, for which #FPT-membership was proved using the compactor enumeration technique, appeared in [60]. Let $p, r \in \mathbb{N}^+$ and let \mathcal{P} be some fixed collection of graphs each of no more than p vertices. We consider the following general problem.

#MINIMUM r -COVERING BY GRAPHS IN \mathcal{P} (in short #MCG(r, \mathcal{P}))

Instance: a graph G , and a $k \in \mathbb{N}$.

Parameter: k .

Question: the number of collections \mathcal{C} of k different subgraphs of G such that

1. each graph in \mathcal{C} is a graph in \mathcal{P} and
2. each connected component of $G \setminus \bigcup_{H \in \mathcal{C}} V(H)$ has size at most r .

If in the conditions of the above problem we add a third one demanding that the graphs in \mathcal{C} are pairwise disjoint, then we define the problem #MINIMUM r -MAXIMAL MATCHING BY GRAPHS IN \mathcal{P} (in short #MMM(r, \mathcal{P})).

#MCG(r, \mathcal{P}) can be seen as a generalization of the the # p -VERTEX COVER problem. To see this, take $\mathcal{P} = \{K_1\}$ and $r = 1$. Also, #MMM(r, \mathcal{P}) can be seen as a generalization of the the # p -MINIMUM MAXIMAL MATCHING problem (take $\mathcal{P} = \{K_2\}$ and $r = 1$). According to [60], both #MCG(r, \mathcal{P}) and #MMM(r, \mathcal{P}) belong to #FPT because of FPT-algorithms running in $O(n \cdot r(pk + r) + 2^{O(pkr(pk+r))})$ steps. The approach of [60] builds a subgraph G' of G on $2^{O(r(pk+r))}$ vertices. Moreover, the vertices of G' are equipped with some extra numeric information on the number of possible solutions of the initial instance that they represent. G' along with the accompanying information can be seen as a compactor of size $2^{O(r(pk+r))}$ that permits, by enumerating all solutions in G' , to reproduce the number of solutions in G .

Both examples of this subsection can be seen as vast extensions of Buss' algorithm. However, the generality of these two problems does not lead to a polynomial size compactor while the algorithm Subsection 1.4 can indeed be seen as a compactor for # p -VERTEX COVER of quadratic size. Moreover, the technique of compactor enumeration as presented in [27, 29] and [60] was not accompanied by the explicit definition of a data-reduction concept where, on the top of this, one could also demand that the reduced instances are of polynomial size. As a viable alternative to that, Marc Thurley introduced,

for the first time in [66, 67], a formal notion of *counting kernelization* and gave counting kernels for a for a series of parameterized counting problems, such as $\#p$ -VERTEX COVER, $\#\text{card-}p$ -HITTING SET, $\#p$ -UNIQUE HITTING SET. We prefer not give the definition of counting kernelization or/and make a comparison between compactors and counting kernels as this escapes the purpose of this survey.

3. Algorithmic meta-theorems on compactors

Algorithmic meta-theorems typically provide general conditions for a problem to admit an efficient algorithm [49, 48, 55, 61]. Typically such conditions are of logical and/or combinatorial nature. Important algorithmic meta-theorems for parameterized problems concern model-checking for Monadic Second Order Logic (MSOL) [13, 8, 3, 62] on bounded treewidth graphs and model checking for First Order Logic (FOL) on certain classes of sparse graphs [39, 47, 23, 38, 37, 50]. Typically, algorithmic meta-theorems encompass the abstraction of what makes a general idea applicable to many problems and reveal deep relations between algorithm theory, logic, and combinatorics.

The purpose of this section is to present an algorithmic meta-theorem (Theorem 3) on the existence of polynomial compactors for a wide family of problems when they are restricted to sparse families of graphs. This theorem appeared in [54] and builds on earlier meta-algorithmic techniques that appeared in [7, 45, 44, 53]. We also present some applications of the main result [54] to general families of problems.

3.1. Monadic Second Order Logic.

The syntax of Monadic Second Order Logic (MSO) on graphs includes the logical connectives \vee , \wedge , \neg , \leftrightarrow , \rightarrow , variables for vertices, edges, sets of vertices, and sets of edges, the quantifiers \forall , \exists that can be applied to these variables and the following predicates:

1. $u \in U$ where u is a vertex variable and U is a vertex set variable;
2. $e \in D$ where e is an edge variable and D is an edge set variable;
3. $e \dashv\circ u$ where e is an edge variable, u is a vertex variable, and the interpretation is that the edge e is incident with the vertex u ;
4. $u \sim v$ where u and v are vertex variables and the interpretation is that u and v are adjacent;

5. equality (“=”) of variables representing vertices, edges, sets of vertices, and sets of edges.

We consider MSOL-sentences with one free vertex-set variable. Given such a sentence ϕ and an annotated graph (G, S) , we write $(G, S) \models \phi$ in order to denote that the pair (G, S) is a model for the sentence ϕ . We correspond each such ϕ to a set of annotated graphs as follows

$$\mathcal{A}_\phi = \{(G, A) \mid (G, A) \models \phi\}.$$

We write $\#\Pi_\phi = (F_\phi, \kappa)$ as a shortcut for $\#\Pi_{\mathcal{A}_\phi} = (F_{\mathcal{A}_\phi}, \kappa)$. We say that an instance $\langle G, k \rangle$ of $\#\Pi_\phi$ is a *null* instance if $F_\phi(G, k) = 0$, i.e., there is no $S \in \binom{V(G)}{k}$ such that $(G, S) \models \phi$. A parameterized counting problem $\#\Pi$ is *MSOL-expressible* if there exists a MSOL-sentence with one free vertex-set variable such that $\#\Pi = \#\Pi_\phi$.

We conclude this subsection by expressing some of the problems of Subsection 1.2 using MSOL-sentences. In the sentences below, symbols v, u, x, y, z represent vertices, e represents an edge, and S, S' represent vertex sets.

- **#p-VERTEX COVER:** $\#\Pi_{\text{vc}} = \#\Pi_{\phi_{\text{vc}}}$ where

$$\phi_{\text{vc}} = \forall e \exists v (v \in S \wedge e \rightarrow v).$$

- **#p-DOMINATING SET:** $\#\Pi_{\text{pds}} = \#\Pi_{\phi_{\text{pds}}}$ where

$$\phi_{\text{pds}} = \forall v (v \in S \vee (\exists u u \in S \wedge v \sim u))$$

- **#p-FEEDBACK VERTEX SET:** $\#\Pi_{\text{fvs}} = \#\Pi_{\phi_{\text{fvs}}}$ where

$$\phi_{\text{fvs}} = \forall S' (\mathbf{cycl}(S') \rightarrow \exists v v \in S \wedge v \in S') \text{ and}$$

$\mathbf{cycl}(S') = \forall x (x \in S' \rightarrow \exists y \exists z (y \in S' \wedge z \in S' \wedge \neg(x = y) \wedge \neg(x = z) \wedge \neg(y = z) \wedge x \sim y \wedge x \sim z))$, i.e., the predicate $\mathbf{cycl}(S')$ expresses the fact that $G[S']$ contains a cycle.

- **#p-CYCLE DOMINATION:** $\#\Pi_{\text{cd}} = \#\Pi_{\phi_{\text{cd}}}$ where

$$\phi_{\text{cd}} = \forall S' (\mathbf{cycl}(S') \rightarrow \exists v (v \in S \wedge (v \in S' \vee (\exists u u \sim v \wedge u \in S')))).$$

- $\#p$ -INDEPENDENT SET: $\#\Pi_{\text{is}} = \#\Pi_{\phi_{\text{is}}}$ where

$$\phi_{\text{is}} = \forall e \neg(\exists v \exists u (v \in S \wedge u \in S \wedge e \dashv\vdash v \wedge e \dashv\vdash u)).$$

We avoid the description of a MSOL-sentence for $\#p$ -LONGEST CYCLE as it is too technical.

Notice that the planar restriction of each of the above problems is also MSOL-expressible, because planarity is a property that can be expressed by a predicate in MSO logic. In general, if \mathcal{Z} is a MSOL-expressible graph class and Π is a MSOL-expressible problem, then $\Pi \pitchfork \mathcal{Z}$ is also MSOL-expressible [7]. Keep in mind that for every finite set of graphs \mathcal{H} , both $\mathcal{M}_{\mathcal{H}}$ and $\mathcal{T}_{\mathcal{H}}$ are MSOL-expressible.

3.2. Treewidth-modulators

Treewidth.. Given a $k \in \mathbb{N}^+$, we say that a graph G is a k -tree if G is isomorphic to K_{k+1} or (recursively) there is a vertex v in G where $N_G[\{v\}]$ is isomorphic to K_{k+1} and $G \setminus \{v\}$ is a k -tree. The *treewidth* of a G is the minimum k for which G is a subgraph of some k -tree.

Treewidth-modulators.. Given a graph G , we say that a vertex set $S \subseteq V(G)$ is a t -treewidth-modulator of G if the removal of S from G produces a graph of treewidth at most t . Given an MSOL-sentence ϕ , we say that $\#\Pi_{\phi}$ is *treewidth-modulable* if there is a constant t (depending on ϕ) such that, for every non-null instance $\langle G, k \rangle$ of $\#\Pi_{\phi}$, G has a t -treewidth-modulator of size at most $t \cdot k$. Clearly if $\#\Pi_{\phi}$ is treewidth-modulable then $\#\Pi_{\phi} \pitchfork \mathcal{Z}$ is also treewidth-modulable, for every $\mathcal{Z} \subseteq \mathcal{G}$.

As an example, notice that $\#\Pi_{\text{fvs}}$ (that is $\#p$ -FEEDBACK VERTEX SET) is trivially treewidth-modulable because if $F_{\phi_{\text{fvs}}}(G, k) \neq 0$ then G contains some subset S such that $G \setminus S$ is acyclic and S is a 1-treewidth-modulator of G with k vertices (note that the acyclic graphs are exactly those of treewidth at most 1).

A general treewidth-modulable problem.. We now give a general family of parameterized counting problems that are treewidth-modulable. Let \mathcal{H} be a finite set of graphs and $d \in \mathbb{N}$. We define

$$\mathcal{A}_{\mathcal{H}, d} = \{(G, S) \mid G \setminus N_G^{(d)}(S) \text{ does not contain any graph in } \mathcal{H} \text{ as a minor}\}.$$

As graph distance as well as the minor relation can be expressed in MSOL, there is a $\phi_{\mathcal{H},d}$ such that $\#\Pi_{\phi_{\mathcal{H},d}} = \#\Pi_{\mathcal{A}_{\mathcal{H},d}}$. For simplicity we denote $\#\Pi_{\mathcal{H},d} := \#\Pi_{\phi_{\mathcal{H},d}}$.

Notice that $\#\Pi_{\mathcal{H},d}$ is a quite general problem. For instance,

- $\#\Pi_{\{K_2\},0}$ is the $\#p$ -VERTEX COVER problem,
- $\#\Pi_{\{K_3\},0}$ is the $\#p$ -FEEDBACK VERTEX SET problem,
- $\#\Pi_{\{K_5, K_{3,3}\},0}$ is the $\#p$ -VERTEX PLANARIZATION problem, asking for the number of planarizers of G of size k .
- $\#\Pi_{\{K_1\},1}$ is the $\#p$ -DOMINATING SET problem while,
- for $d \geq 2$, $\#\Pi_{\{K_1\},d}$ is the $\#p$ - d -DISTANCE DOMINATION problem².
- $\#\Pi_{\{K_3\},1}$ is the $\#p$ -CYCLE DOMINATION problem.

The proof of the next theorem appeared, in different forms, in [7, 43, 5, 53].

Theorem 1. *If \mathcal{H} is a set of finite connected graphs containing at least one planar graph, then $\#\Pi_{\mathcal{H},0}$ is treewidth-modulable.*

The next theorem follows from the results of [43], taking into account that if \mathcal{H} contains at least one planar graph, then $\Pi_{\mathcal{H},d}$ is contraction-bidimensional, for every $d \in \mathbb{N}$ (see [26, 25, 42, 24, 64, 63] for surveys on Bidimensionality Theory).

Theorem 2. *If \mathcal{H} is a set of finite connected graphs containing at least one planar graph, \mathcal{C} is a set of finite graphs containing at least one apex³ graph, and $d \in \mathbb{N}$, then $\#\Pi_{\mathcal{H},d} \pitchfork \mathcal{M}_{\mathcal{C}}$ is treewidth-modulable.*

²The $\#p$ - d -DISTANCE DOMINATION problem is $\#\Pi_{\mathcal{A}_{ds}^{(d)}}$ where $\mathcal{A}_{ds}^{(d)} = \{(G, S) \mid V(G) = N_G^{(d)}[S]\}$.

³A graph G is an *apex graph* if it contains a planarizer of cardinality 1. Alternatively, G is an apex graph if $\langle G, 1 \rangle$ is a non-null instance of $\#\Pi_{\{K_5, K_{3,3}\},0}$.

An algorithmic meta-theorem for counting parameterized problems.. The next theorem is the main result in [54].

Theorem 3. *For every finite set of graphs \mathcal{C} and every counting parameterized problem $\#\Pi$ that is MSOL-expressible and treewidth-modulable, there is a compactor for $\Pi \pitchfork \mathcal{T}_{\mathcal{C}}$ 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 1. *For every finite set of graphs \mathcal{C} and every counting parameterized problem $\#\Pi$ that is MSOL-expressible and treewidth-modulable, there is an FPT-algorithm that solves $\#\Pi \pitchfork \mathcal{T}_{\mathcal{C}}$ in $O(k^2n^2) + 2^{O(k)}$ steps.*

In the above results, the constants hidden in the O -notation depend on the MSOL-sentence expressing $\#\Pi$, on the treewidth-modulability constant t , and on the choice of \mathcal{C} . The application of the above two results on the treewidth-modulable instantiations of the general problem $\#\Pi_{\mathcal{H},d}$ implies the following.

Corollary 2. *Let \mathcal{H} be a finite set of connected graphs containing at least one planar graph, \mathcal{C} be a finite set of graphs, and $d \in \mathbb{N}$. Suppose also that*

- $\mathcal{Z} := \mathcal{M}_{\mathcal{C}}$ and \mathcal{C} contains at least one apex graph or
- $\mathcal{Z} := \mathcal{T}_{\mathcal{C}}$ and $d = 0$.

Then there is a compactor for $\#\Pi_{\mathcal{H},d} \pitchfork \mathcal{Z}$ of size $O(k^2)$ with condensing time $O(k^2n^2)$ and decoding time $2^{O(k)}$. Moreover there is an FPT-algorithm that solves $\#\Pi_{\mathcal{H},d} \pitchfork \mathcal{Z}$ in $O(k^2n^2) + 2^{O(k)}$ steps.

Acknowledgement.. The author wish to thank EunJung Kim for her advise and her contribution to the results presented in this paper.

Postscript.. This work is dedicated to my beloved [Profesora Maria José Serna Iglesias](#). It is also an opportunity to express my gratitude for all that I learned from her.

References

- [1] Abrahamson, K.A., Downey, R.G., Fellows, M.R., 1995. Fixed-parameter tractability and completeness. IV. On completeness for $W[P]$ and PSPACE analogues. *Annals of Pure and Applied Logic* 73, 235–276.
- [2] Aho, A.V., Hopcroft, J.E., Ullman, J.D., 1974. *The Design and Analysis of Computer Algorithms*. Addison-Wesley.
- [3] Arnborg, S., Lagergren, J., Seese, D., 1991. Easy problems for tree-decomposable graphs. *Journal of Algorithms* 12, 308–340.
- [4] Arvind, V., Raman, V., 2002. Approximation algorithms for some parameterized counting problems, in: *Proc. 13th ISAAC*. Springer. volume 2518 of *LNCS*, pp. 453–464.
- [5] Baste, J., Sau, I., Thilikos, D.M., 2017. Optimal algorithms for hitting (topological) minors on graphs of bounded treewidth, in: *12th International Symposium on Parameterized and Exact Computation, IPEC 2017*, September 6-8, 2017, Vienna, Austria, pp. 4:1–4:12.
- [6] Bläser, M., Curticapean, R., 2012. Weighted counting of k -matchings is $\#W[1]$ -hard, in: Thilikos, D.M., Woeginger, G.J. (Eds.), *Parameterized and Exact Computation - 7th International Symposium, IPEC 2012*, Ljubljana, Slovenia, September 12-14, 2012. *Proceedings*, Springer. pp. 171–181. URL: https://doi.org/10.1007/978-3-642-33293-7_17, doi:10.1007/978-3-642-33293-7_17.
- [7] Bodlaender, H.L., Fomin, F.V., Lokshtanov, D., Penninkx, E., Saurabh, S., Thilikos, D.M., 2016. (meta) kernelization. *J. ACM* 63, 44:1–44:69.
- [8] Borie, R.B., Parker, R.G., Tovey, C.A., 1992. Automatic generation of linear-time algorithms from predicate calculus descriptions of problems on recursively constructed graph families. *Algorithmica* 7, 555–581.
- [9] Brand, C., Roth, M., 2017. Parameterized counting of trees, forests and matroid bases, in: Weil, P. (Ed.), *Computer Science - Theory and Applications - 12th International Computer Science Symposium in Russia, CSR 2017*, Kazan, Russia, June 8-12, 2017, *Proceedings*, Springer. pp. 85–98. URL: https://doi.org/10.1007/978-3-319-58747-9_10, doi:10.1007/978-3-319-58747-9_10.

- [10] Buss, J.F., Goldsmith, J., 1993. Nondeterminism within P. *SIAM J. Comput.* 22, 560–572.
- [11] Chen, J., Fernau, H., Kanj, I.A., Xia, G., 2007. Parametric duality and kernelization: Lower bounds and upper bounds on kernel size. *SIAM J. Comput.* 37, 1077–1106.
- [12] Chen, J., Kanj, I.A., Xia, G., 2010. Improved upper bounds for vertex cover. *Theor. Comput. Sci.* 411, 3736–3756. URL: <http://dx.doi.org/10.1016/j.tcs.2010.06.026>, doi:<http://dx.doi.org/10.1016/j.tcs.2010.06.026>.
- [13] Courcelle, B., 1990. The monadic second-order logic of graphs. I. Recognizable sets of finite graphs. *Information and Computation* 85, 12–75.
- [14] Curticapean, R., 2016a. Counting matchings with k unmatched vertices in planar graphs, in: Sankowski, P., Zaroliagis, C.D. (Eds.), 24th Annual European Symposium on Algorithms, ESA 2016, August 22-24, 2016, Aarhus, Denmark, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik. pp. 33:1–33:17. URL: <https://doi.org/10.4230/LIPIcs.ESA.2016.33>, doi:10.4230/LIPIcs.ESA.2016.33.
- [15] Curticapean, R., 2016b. The simple, little and slow things count: on parameterized counting complexity. *Bulletin of the EATCS* 120. URL: <http://eatcs.org/beatcs/index.php/beatcs/article/view/445>.
- [16] Curticapean, R., 2018. Block interpolation: A framework for tight exponential-time counting complexity. *Inf. Comput.* 261, 265–280. URL: <https://doi.org/10.1016/j.ic.2018.02.008>, doi:10.1016/j.ic.2018.02.008.
- [17] Curticapean, R., Dell, H., Marx, D., 2017a. Homomorphisms are a good basis for counting small subgraphs, in: Hatami, H., McKenzie, P., King, V. (Eds.), Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017, ACM. pp. 210–223. URL: <https://doi.org/10.1145/3055399.3055502>, doi:10.1145/3055399.3055502.
- [18] Curticapean, R., Dell, H., Roth, M., 2017b. Counting edge-injective homomorphisms and matchings on restricted graph classes,

- in: Vollmer, H., Vallée, B. (Eds.), 34th Symposium on Theoretical Aspects of Computer Science, STACS 2017, March 8-11, 2017, Hannover, Germany, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik. pp. 25:1–25:15. URL: <https://doi.org/10.4230/LIPIcs.STACS.2017.25>, doi:10.4230/LIPIcs.STACS.2017.25.
- [19] Curticapean, R., Lindzey, N., Nederlof, J., 2018. A tight lower bound for counting hamiltonian cycles via matrix rank, in: Czumaj, A. (Ed.), Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018, SIAM. pp. 1080–1099. URL: <https://doi.org/10.1137/1.9781611975031.70>, doi:10.1137/1.9781611975031.70.
- [20] Curticapean, R., Marx, D., 2014. Complexity of counting subgraphs: Only the boundedness of the vertex-cover number counts, in: 55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014, IEEE Computer Society. pp. 130–139. URL: <https://doi.org/10.1109/FOCS.2014.22>, doi:10.1109/FOCS.2014.22.
- [21] Curticapean, R., Marx, D., 2016. Tight conditional lower bounds for counting perfect matchings on graphs of bounded treewidth, cliquewidth, and genus, in: Krauthgamer, R. (Ed.), Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016, SIAM. pp. 1650–1669. URL: <https://doi.org/10.1137/1.9781611974331.ch113>, doi:10.1137/1.9781611974331.ch113.
- [22] Cygan, M., Fomin, F.V., Kowalik, L., Lokshtanov, D., Marx, D., Pilipczuk, M., Pilipczuk, M., Saurabh, S., 2015. Parameterized Algorithms. Springer.
- [23] Dawar, A., Grohe, M., Kreutzer, S., 2007. Locally excluding a minor, in: 21st IEEE Symposium on Logic in Computer Science (LICS'07). IEEE, New York, pp. 270–279.
- [24] Demaine, E., 2010. Algorithmic graph minors and bidimensionality, in: Thilikos, D. (Ed.), Graph Theoretic Concepts in Computer Science. Springer Berlin / Heidelberg. volume 6410 of *Lecture Notes in Computer Science*, pp. 2–2.

- [25] Demaine, E.D., Hajiaghayi, M., 2008a. Bidimensionality, in: Kao, M.Y. (Ed.), *Encyclopedia of Algorithms*. Springer.
- [26] Demaine, E.D., Hajiaghayi, M., 2008b. The bidimensionality theory and its algorithmic applications. *Comput. J.* 51, 292–302.
- [27] Díaz, J., Serna, M., Thilikos, D.M., 2004. Fixed parameter algorithms for counting and deciding bounded restrictive list H -colorings (extended abstract), in: *Algorithms—ESA 2004*. Springer, Berlin. volume 3221 of *LNCS*, pp. 275–286.
- [28] Díaz, J., Serna, M., Thilikos, D.M., 2005. The restrictive H -coloring problem. *Discrete Appl. Math.* 145, 297–305. URL: <http://dx.doi.org/10.1016/j.dam.2004.01.018>, doi:10.1016/j.dam.2004.01.018.
- [29] Díaz, J., Serna, M., Thilikos, D.M., 2008. Efficient algorithms for counting parameterized list H -colorings. *J. Comput. System Sci.* 74, 919–937. URL: <http://dx.doi.org/10.1016/j.jcss.2008.02.004>, doi:10.1016/j.jcss.2008.02.004.
- [30] Díaz, J., Serna, M.J., Thilikos, D.M., 2001a. (H, C, K) -coloring: Fast, easy, and hard cases, in: Sgall, J., Pultr, A., Kolman, P. (Eds.), *Mathematical Foundations of Computer Science 2001, 26th International Symposium, MFCS 2001 Mariánské Lázně, Czech Republic, August 27-31, 2001, Proceedings*, Springer. pp. 304–315. URL: https://doi.org/10.1007/3-540-44683-4_27, doi:10.1007/3-540-44683-4_27.
- [31] Díaz, J., Serna, M.J., Thilikos, D.M., 2001b. Recent results on parameterized H -colorings, in: Nešetřil, J., Winkler, P. (Eds.), *Graphs, Morphisms and Statistical Physics, Proceedings of a DIMACS Workshop, New Brunswick, New Jersey, USA, March 19-21, 2001, DIMACS/AMS*. pp. 65–86. URL: <http://dimacs.rutgers.edu/Volumes/Vol163.html>.
- [32] Díaz, J., Serna, M.J., Thilikos, D.M., 2007. Complexity issues on bounded restrictive H -coloring. *Discrete Mathematics* 307, 2082–2093. URL: <https://doi.org/10.1016/j.disc.2005.12.058>, doi:10.1016/j.disc.2005.12.058.
- [33] Downey, R., Fellows, M., 1993. Fixed-parameter tractability and completeness. III. Some structural aspects of the W hierarchy, in: *Complexity theory*. Cambridge Univ. Press, Cambridge, pp. 191–225.

- [34] Downey, R.G., Fellows, M.R., 1995a. Fixed-parameter tractability and completeness. I. Basic results. *SIAM J. Comput.* 24, 873–921.
- [35] Downey, R.G., Fellows, M.R., 1995b. Fixed-parameter tractability and completeness II: On completeness for $W[1]$. *Theoretical Computer Science* 141, 109–131.
- [36] Downey, R.G., Fellows, M.R., 2013. *Fundamentals of Parameterized Complexity*. Texts in Computer Science, Springer. URL: <http://dx.doi.org/10.1007/978-1-4471-5559-1>, doi:10.1007/978-1-4471-5559-1.
- [37] Dvorak, Z., Kral, D., Thomas, R., 2010. Deciding first-order properties for sparse graphs, in: 2010 IEEE 51st Annual Symposium on Foundations of Computer Science. IEEE, Washington, DC, USA. FOCS '10, pp. 133–142. doi:<http://dx.doi.org/10.1109/FOCS.2010.20>.
- [38] Dvořák, Z., Král, D., Thomas, R., 2013. Testing first-order properties for subclasses of sparse graphs. *J. ACM* 60, 36:1–36:24. URL: <http://doi.acm.org/10.1145/2499483>, doi:10.1145/2499483.
- [39] Flum, J., Grohe, M., 2001. Fixed-parameter tractability, definability, and model-checking. *SIAM J. Comput.* 31, 113–145.
- [40] Flum, J., Grohe, M., 2002. The parameterized complexity of counting problems, in: 43rd Symposium on Foundations of Computer Science. IEEE Computer Society, Washington, DC, USA. FOCS '02, pp. 538–. URL: <http://portal.acm.org/citation.cfm?id=645413.652181>.
- [41] Flum, J., Grohe, M., 2006. *Parameterized Complexity Theory*. Texts in Theoretical Computer Science. An EATCS Series, Springer-Verlag, Berlin.
- [42] Fomin, F.V., Demaine, E.D., Hajiaghayi, M.T., 2015. Bidimensionality, in: *Encyclopedia of Algorithms*. Springer.
- [43] Fomin, F.V., Lokshtanov, D., Misra, N., Saurabh, S., 2012. 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, pp. 470–479.

- [44] Fomin, F.V., Lokshтанov, D., Saurabh, S., Thilikos, D.M., 2010. Bidimensionality and kernels, in: 21st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2010). ACM-SIAM, pp. 503–510.
- [45] Fomin, F.V., Lokshтанov, D., Saurabh, S., Thilikos, D.M., 2016. Bidimensionality and kernels. CoRR abs/1606.05689. URL: <http://arxiv.org/abs/1606.05689>, arXiv:1606.05689. revised version.
- [46] Fomin, F.V., Lokshтанov, D., Saurabh, S., Zehavi, M., 2019. Kernelization: Theory of Parameterized Preprocessing. Cambridge University Press. doi:10.1017/9781107415157.
- [47] Frick, M., Grohe, M., 1999. Deciding first-order properties of locally tree-decomposable graphs, in: Wiedermann, J., van Emde Boas, P., Nielsen, M. (Eds.), Automata, Languages and Programming. Springer Berlin / Heidelberg. volume 1644 of *LNCS*, pp. 72–72.
- [48] Grohe, M., 2007. Logic, Graphs, and Algorithms. Amsterdam University Press. chapter in J.Flum, E.Grädel, T.Wilke (Eds), Logic and Automata — History and Perspectives. pp. 357 – 422.
- [49] Grohe, M., Kreutzer, S., 2011. Methods for algorithmic meta theorems. Contemporary Mathematics. chapter Model Theoretic Methods in Finite Combinatorics. pp. 181 – 206.
- [50] Grohe, M., Kreutzer, S., Siebertz, S., 2014. Deciding first-order properties of nowhere dense graphs, in: Proceedings of the 46th Annual ACM Symposium on Theory of Computing, ACM, New York, NY, USA. pp. 89–98.
- [51] Jerrum, M., Meeks, K., 2015. The parameterised complexity of counting connected subgraphs and graph motifs. *J. Comput. Syst. Sci.* 81, 702–716.
- [52] Jerrum, M., Meeks, K., 2017. The parameterised complexity of counting even and odd induced subgraphs. *Combinatorica* 37, 965–990.
- [53] Kim, E.J., Langer, A., Paul, C., Reidl, F., Rossmanith, P., Sau, I., Sikdar, S., 2016. Linear kernels and single-exponential algorithms via protrusion decompositions. *ACM Trans. Algorithms* 12, 21:1–21:41.

- [54] Kim, E.J., Serna, M.J., Thilikos, D.M., 2018. Data-compression for parametrized counting problems on sparse graphs, in: Hsu, W., Lee, D., Liao, C. (Eds.), 29th International Symposium on Algorithms and Computation, ISAAC 2018, December 16-19, 2018, Jiaoxi, Yilan, Taiwan, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik. pp. 20:1–20:13. URL: <https://doi.org/10.4230/LIPIcs.ISAAC.2018.20>, doi:10.4230/LIPIcs.ISAAC.2018.20.
- [55] Kreutzer, S., 2009. Algorithmic meta-theorems. *Electronic Colloquium on Computational Complexity (ECCC)* 16, 147.
- [56] McCartin, C., 2006. Parameterized counting problems. *Ann. Pure Appl. Logic* 138, 147–182.
- [57] Montoya, J.A., 2011. On the parameterized complexity of approximate counting. *RAIRO - Theor. Inf. and Applic.* 45, 197–223. URL: <https://doi.org/10.1051/ita/2011007>, doi:10.1051/ita/2011007.
- [58] Nemhauser, G.L., Jr., L.E.T., 1975. Vertex packings: structural properties and algorithms. *Math. Programming* 8, 232–248.
- [59] Niedermeier, R., 2002. Invitation to fixed-parameter algorithms. Ph.D. thesis. Universität Tübingen. Habilitation thesis.
- [60] Nishimura, N., Ragde, P., Thilikos, D.M., 2005. Parameterized counting algorithms for general graph covering problems, in: Dehne, F.K.H.A., López-Ortiz, A., Sack, J. (Eds.), *Algorithms and Data Structures*, 9th International Workshop, WADS 2005, Waterloo, Canada, August 15-17, 2005, Proceedings, Springer. pp. 99–109. URL: https://doi.org/10.1007/11534273_10, doi:10.1007/11534273_10.
- [61] Seese, D., 1991. The structure of the models of decidable monadic theories of graphs. *Annals of Pure and Applied Logic* 53, 169 – 195. URL: <http://www.sciencedirect.com/science/article/pii/016800729190054P>, doi:[https://doi.org/10.1016/0168-0072\(91\)90054-P](https://doi.org/10.1016/0168-0072(91)90054-P).
- [62] Seese, D., 1996. Linear time computable problems and first-order descriptions. *Mathematical Structures in Computer Science* 6, 505–526.

- [63] Thilikos, D.M., 2012. Graph minors and parameterized algorithm design, in: *The Multivariate Algorithmic Revolution and Beyond - Essays Dedicated to Michael R. Fellows on the Occasion of His 60th Birthday*, pp. 228–256.
- [64] Thilikos, D.M., 2015. Bidimensionality and parameterized algorithms (invited talk), in: *10th International Symposium on Parameterized and Exact Computation, IPEC 2015, September 16-18, 2015, Patras, Greece*, pp. 1–16.
- [65] Thomassé, S., 2010. A $4k^2$ kernel for feedback vertex set. *ACM Trans. Algorithms* 6, 32:1–32:8. URL: <http://doi.acm.org/10.1145/1721837.1721848>, doi:10.1145/1721837.1721848.
- [66] Thurley, M., 2007. Kernelizations for parameterized counting problems, in: *4th international conference on Theory and applications of models of computation. Springer-Verlag, Berlin, Heidelberg. TAMC'07*, pp. 705–714. URL: <http://portal.acm.org/citation.cfm?id=1767854.1767920>.
- [67] Thurley, M., May 2006. Tractability and Intractability of Parameterized Counting Problems. Master's thesis. Humboldt Universität zu Berlin. Diploma thesis.