Fast Parallel Fixed-parameter Algorithms via Color Coding*

Max Bannach[1,2], Christoph Stockhusen[1], and Till Tantau[1]

1 **Institute for Theoretical Computer Science**
Universität zu Lübeck
Lübeck, Germany
{bannach,stockhus,tantau}@tcs.uni-luebeck.de
2 **Graduate School for Computing in Medicine and Life Sciences**
Universität zu Lübeck, Germany

Abstract

Fixed-parameter algorithms have been successfully applied to solve numerous difficult problems within acceptable time bounds on large inputs. However, most fixed-parameter algorithms are inherently *sequential* and, thus, make no use of the parallel hardware present in modern computers. We show that parallel fixed-parameter algorithms do not only exist for numerous parameterized problems from the literature – including vertex cover, packing problems, cluster editing, cutting vertices, finding embeddings, or finding matchings – but that there are parallel algorithms working in *constant* time or at least in time *depending only on the parameter* (and not on the size of the input) for these problems. Phrased in terms of complexity classes, we place numerous natural parameterized problems in parameterized versions of AC^0. On a more technical level, we show how the *color coding* method can be implemented in constant time and apply it to embedding problems for graphs of bounded tree-width or tree-depth and to model checking first-order formulas in graphs of bounded degree.

1998 ACM Subject Classification F.1.3 Complexity Measures and Classes

Keywords and phrases color coding, parallel computation, fixed-parameter tractability, graph packing, cutting ℓ vertices, cluster editing, tree-width, tree-depth, model checking

Digital Object Identifier 10.4230/LIPIcs.IPEC.2015.224

1 Introduction

The classical objective of parameterized complexity theory is to determine for a parameterized problem whether it can be solved by an algorithm running in time $f(k) \cdot n^c$, where f is some function, k is a parameter, n is the input length, and c is some constant. Such algorithms are nowadays routinely used to solve large instances for NP- or even PSPACE-hard problems within acceptable amounts of time. Nevertheless, "acceptable" is not the same as "small" and one would like to further reduce the runtime by using multiple cores to speed up the computation. For this, one needs *parallel* fixed-parameter algorithms, but most fixed-parameter algorithms have been devised with a sequential computation model in mind. Indeed, the most important tool of parameterized complexity theory, namely kernelization, is inherently sequential: It asks us to repeatedly apply rules to an input, each time modifying the input slightly and making it a little smaller, until the input's size only depends on the parameter. There is no straightforward way of parallelizing such algorithms since later

* A full version of this paper is available under http://arxiv.org/abs/1509.06984.

10th International Symposium on Parameterized and Exact Computation (IPEC 2015).
Editors: Thore Husfeldt and Iyad Kanj; pp. 224–235

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

modifications strongly depend on what happened earlier, forcing us to apply the typically very large number of kernelization steps in a sequential manner.

Our Contributions. The purpose of the present paper is to show that not only do parallel fixed-parameter algorithms exist for many natural, well-studied problems from the literature; for certain problems there are even parallel algorithms that require only *constant* time in a concurrent-read, concurrent-write PRAM model (so the runtime is totally independent of the input) or at least time *depending only on the parameter* (so the length of the input is irrelevant). In all cases, the *work* done by the algorithms is still $f(k) \cdot n^c$, that is, the same as the time bound for sequential fixed-parameter algorithms.[1] Phrased more formally, our objective is to identify parameterized problems that lie in the complexity classes para-$\mathrm{AC}_{O(1)}$ and para-$\mathrm{AC}_{f(k)}$ (formal definitions will be given later).

In order to tackle the parallel parameterized complexity of natural problems like the vertex cover problem, we introduce three technical tools. The first and foremost is *color coding*: all of our proofs employ this technique at least indirectly and we show that the *universal coloring families* that lie at the heart of the technique can be computed in constant time. Second, numerous natural "packing problems" are special cases of the following embedding problem: Given graphs H and G, find a (not necessarily induced) subgraph of G that is isomorphic to H. We give new bounds on the complexity of this problem when H has bounded tree-width or bounded tree-depth; and these bounds later translate directly to bounds on different packing problems. Third, we translate an algorithmic meta-theorem of Flum and Grohe [16] to the parallel world: We show that model checking first-order properties of graphs can be done in parallel in time depending only on the parameters (actually, only on the locality rank of the formula), where the parameters are the to-be-checked formula and the degree of the graph.

We then apply the tools to a wide variety of natural graph problems, namely *packing problems, covering problems, clustering problems,* and *separation problems.* For *packing problems* the objective is to determine whether a given graph G contains k vertex-disjoint copies of some fixed graph H like, say, a triangle. Even for triangles, this problem is already NP-complete, but when k is considered as a parameter, the triangle packing problem lies in FPT [15]. We show that there is a constant-time, FPT-work algorithm for triangle packing – and indeed for packing any graph of fixed size. The *covering problems* we study include the vertex cover problem and its partial version. We present a constant parallel time algorithm for the first problem and an algorithm for the second needing time depending only on the parameter. These results nicely reflect on a theoretical basis the "empirical" observation that p-VERTEX-COVER is one of the "easiest" parameterized problems and that the partial version is a bit harder to solve. For *clustering problems,* also known as *cluster editing problems,* the objective is to transform a graph by adding or deleting few edges into a collection of "clusters" – which are just cliques in the simplest case. We present a constant time, FPT-work algorithm for cluster editing. For *graph separation problems* the objective is to "cut away" a special part of a graph using few vertices. We show that certain versions of these problems can be solved by a parallel fixed-parameter algorithm in time depending only on the parameter and FPT work (while other versions are known to be W[1]-hard).

[1] The *work* done by a parallel algorithm is the total number of computational steps made by all computational units during a computation. Since "all work needs to be done," in practice the runtime of a parallel algorithm is its work divided by the number of available cores. In particular, the *work* done by a *parallel algorithm* should not exceed the *runtime* of a *sequential algorithm* for the same problem. In our case, this means that in order to compete with sequential algorithms running in "FPT time," our parallel algorithm must not only be fast, but may only do "FPT work."

Related Work. There is a growing body of literature reporting on the practicalities of implementing fixed-parameter algorithms in parallel [1]. In contrast, there are only few results addressing parallel fixed-parameter tractability on a theoretical level (as we do in the present paper), see for instance Cesati and Di Ianni [9]. Since it is well-known from classical complexity theory that problems solvable in logarithmic space can be parallelized well, previous research on *parameterized logarithmic space* contributes to our understanding of which parameterized problems can be parallelized in principle. This research was started by Cai, Chen, Downey, and Fellows [8]. First (quite technical) complete problems for parameterized logarithmic space where later introduced by Chen, Flum, and Grohe [11], and by Flum and Grohe [16]. A more structural study of parameterized space and circuit classes (which addresses parallelization more directly) was later made by Elberfeld and the last two authors [14]. Parameterized Circuit Complexity was also studied by Downey et al. with respect to the Weft Hierarchy [13]. Recently, Chen and Müller [10] connected color coding and parameterized space in an algorithm for finding embedding of bounded tree-depth graphs in parameterized logarithmic space (a result which we strengthen considerably in Corollary 3.7).

The first use of the color coding technique can be traced back to Alon, Yuster, and Zwick [2]. They used the technique to provide an FPT-algorithm that decides whether there is an embedding of a graph H of bounded tree-*width* into another graph G, where H is the parameter.

Organization of This Paper. In Section 2 we give formal definitions of the classes of problems solvable by parallel fixed-parameter algorithms. While most of our definitions and classes are standard, the class of problems solvable in "time depending on the parameter and FPT work" seems to be new. In Section 3 we introduce our three technical tools – color coding, embeddings, and model checking – and prove the results mentioned earlier. In Section 4 we study the complexity of the natural parameterized graph problems and establish new upper bounds on their complexities. Due to lack of space, most proofs are only available in the full version; we give proof sketches for some of them in the main text.

2 Classes of Fixed-parameter Parallelism

For our definition of parallel fixed-parameter tractability, we mostly use the standard terminology of parameterized complexity theory, see for instance [17]: A *parameterized problem* is a tuple (Q, κ) of a language $Q \subseteq \Sigma^*$ over an alphabet Σ and a parameterization $\kappa\colon \Sigma^* \to \mathbb{N}$ that maps instances to parameter values. In the classical definition, Downey and Fellows [12] require the parameterization to be computable, while Flum and Grohe [17] require it to be computable in polynomial time. Elberfeld and the last two authors require it to be computable in logarithmic space [14] and mention that it would be better if the parameterization is first-order computable (FO-computable) or, equivalently, to be computable by logarithmic-time-uniform constant depth circuits [24]. Since we will only deal with parameterized circuit classes that lie within parameterized logarithmic space, we will require all parameterizations to be FO-computable. We denote parameterized problems with a leading "p-" as in p-VERTEX-COVER and, when the parameter may be unclear, add it as an index as in $p_{|H|}$-EMB.

A parameterized problem is *fixed-parameter tractable* if it can be decided in time $f(\kappa(x)) \cdot |x|^c$ for any input x, where f is some computable function and c a constant. An equivalent definition is that there exists a set $R \in \mathrm{P}$, where P denotes the class of languages decidable in

polynomial time, such that $x \in Q$ iff $\left(x, 1^{f(\kappa(x))}\right) \in R$. The first definition of fixed-parameter tractability gave rise to the class name FPT in the literature, while the second definition gives rise to the name para-P for the same class. The advantage of the second definition is that we can replace the class P in the definition by arbitrary complexity classes and arrive at classes like *parameterized logarithmic space,* para-L, or *parameterized constant depth circuits*, para-AC^0. These parameterized classes inherit their inclusion structure from the classical classes, so we have

$$\text{para-AC}^0 \subsetneq \text{para-TC}^0 \subseteq \text{para-NC}^1 \subseteq \text{para-L} \subseteq \text{para-NL} \subseteq \text{para-AC}^1 \subseteq \text{para-P}.$$

It is not quite obvious, but the class para-AC^0 already captures one of the types of algorithms mentioned in the introduction, namely "constant time, FPT-work," while none of the above classes seems to capture "parameter time, FPT-work." For this reason and in order to explicitly spell out what para-AC^0 contains, we provide a new definition:[2]

▶ **Definition 2.1** (Classes of Parallel Fixed-Parameter Tractability)**.** Let $d\colon \mathbb{N}^2 \to \mathbb{N}$ be a *depth bounding function* and $w\colon \mathbb{N}^2 \to \mathbb{N}$ be a *width bounding function* which both map each pair of an input length and a parameter to a number. We define para-$\mathrm{AC}[d, w]$ as the class of parameterized problems (Q, κ) for which there exists a DLOGTIME-uniform[3] family $(C_{n,k})_{n,k\in\mathbb{N}}$ of AC-circuits (only NOT-, AND-, and OR-gates are allowed, AND- and OR- gates may have unbounded fan-in) such that:

1. For all $x \in \Sigma^*$, the circuit $C_{|x|,\kappa(x)}$ evaluates to 1 on input x if, and only if, $x \in Q$.
2. The depth of each $C_{n,k}$ is at most $d(n, k)$.
3. The size of each $C_{n,k}$ is at most $w(n, k)$.

In the present paper we exclusively study parallel algorithms with "FPT-work" and are therefore only interested in the case where w is member of the family W of functions of the form $f(k) \cdot n^c$ for a computable function f and a constant c. We introduce for arbitrary families D of functions $d\colon \mathbb{N}^2 \to \mathbb{N}$ the abbreviation para-AC_D for $\bigcup_{d\in D, w\in W}$ para-$\mathrm{AC}[d, w]$. For constant depth bounding functions the resulting class para-$\mathrm{AC}_{O(1)}$ is the same as the class para-AC^0.[4] For arbitrary $i > 0$ we obtain para-$\mathrm{AC}_{\{\, f(k)+c\cdot\log^i n \mid f\colon \mathbb{N}\to\mathbb{N}\wedge c\in\mathbb{N}\,\}}$ = para-AC^i(in slight abuse of notation we will write such classes simply as para-$\mathrm{AC}_{f(k)+O(\log^i n)}$).

When the depth bounding function just depends on the parameter, so $d(n, k) = f(k)$, we get a new class para-$\mathrm{AC}_{f(k)}$ that we abbreviate with para-$\mathrm{AC}^{0\uparrow}$. This class does not seem to arise from substituting some classical class for P in the definition of para-P. In particular, this class seems to be incomparable with all classes between para-TC^0 and para-NL. It is, however, clearly contained in para-AC^1, and is strictly more powerful then para-AC^0 as we will see later. This class captures the problems solvable in "parameter time, FPT-work" and we have

$$\text{para-AC}^0 \subsetneq \text{para-AC}^{0\uparrow} \subseteq \text{para-AC}^1.$$

Let us define for arbitrary $i \geq 0$ the class para-$\mathrm{AC}^{i\uparrow}$ as para-$\mathrm{AC}_{f(k)\cdot O(\log^i n)}$. Notice that we have by definition the inclusion structure para-$\mathrm{AC}^i \subseteq$ para-$\mathrm{AC}^{i\uparrow} \subseteq$ para-$\mathrm{AC}^{i+\epsilon}$.

---

[2] The definition can trivially be adjusted to use TC-circuits or NC-circuits, but we will not need them.

[3] We use DLOGTIME-uniform families since they are equivalent to first-order definable families and constitute one of the strongest forms of uniformity [4].

[4] Since the designation para-AC^0 has been used in previous publications and is a bit shorter, we will use it in the following.

3 Technical Tools

3.1 Color Coding in Constant Parallel Time

The idea of color coding is best understood by a concrete application, for instance to the well-known matching problem: Given an undirected graph G and a number k, does G contain k edges such that no two of them share any endpoints? Directly solving this problem is not easy since the known polynomial-time algorithms for it are rather involved. Consider, however, what happens when we randomly color the graph with k colors and then check whether *the vertices of each color class contain at least one edge*. Clearly, if this is the case, there is a matching of size k – and if there is no such matching, then no coloring will pass the test.

We now formalize the idea behind color coding and then show how the colorings can be computed in constant time. It turns out that one can derandomize the computation of a coloring: instead of random colorings we use sets of colorings such that for every set of k vertices and "desired" colors for them, at least one coloring colors the vertices as desired:

▶ **Definition 3.1** (Universal Coloring Families)**.** For natural numbers n, k, and c, an (n, k, c)-*universal coloring family* is a set Λ of functions $\lambda\colon \{1, \dots, n\} \to \{1, \dots, c\}$ such that for every subset $S \subseteq \{1, \dots, n\}$ of size $|S| = k$ and for every mapping $\mu\colon S \to \{1, \dots, c\}$ there is at least one function $\lambda \in \Lambda$ with $\forall s \in S\colon \mu(s) = \lambda(s)$.

The matching problem can be solved easily when we have access to a $(n, 2k, k)$-universal coloring family: If there is a matching of size k, the family will contain some coloring that colors the two endpoints of the first edge with color 1, the endpoints of the second edge with color 2, and so on. Thus there is, indeed, a matching of size k in the graph if for at least one coloring every color class contains an edge. Since we can easily check in parallel for all colorings whether this is the case for one of them, the complexity of p_k-MATCHING hinges critically on the complexity of computing the universal coloring family and the size of this family. The next theorem shows that (n, k, c)-universal coloring families of reasonable size can be computed "in constant time and work $f(k, c) \cdot n^{O(1)}$," which implies that p_k-MATCHING $\in$ para-AC^0 holds:

▶ **Theorem 3.2.** *There is a* DLOGTIME*-uniform family* $(C_{n,k,c})_{n,k,c\in\mathbb{N}}$ *of* AC*-circuits without inputs such that each* $C_{n,k,c}$

1. *outputs an* (n, k, c)*-universal coloring family (coded as a sequence of function tables),*
2. *has constant depth (independent of* n*,* k*, or* c*), and*
3. *has size at most* $O(n \log c \cdot c^{k^2} \cdot k^4 \log^2 n)$*.*

Sketch of Proof. The family of universal coloring functions we construct is based on the concept of k-perfect hash functions [17], that, after slight modifications, provide us with the desired coloring properties. The crucial part is to implement them using circuits that are DLOGTIME-uniform. However, we can achieve this, since the numbers n, k, and c are encoded in unary and the operations required to compute the functions are only additions, multiplications, and modulo operations. ◀

Investigating a parameterized version of matching may seem a bit strange at first sight – matching is even known to be solvable in randomized polylogarithmic parallel time. However, the exact parallel time complexity is still open in the classical setting while from a parameterized perspective, we just saw that the matching can be solved *very* quickly in parallel. Another problem that one would maybe not expect to be studied in the parameterized

setting, but which will be useful in a number of situations, is p-THRESHOLD. The inputs are a bitstring $b \in \{0,1\}^n$ and a parameter t. The question is whether there are at least t many 1's in b. Clearly, the unparameterized version is complete for TC^0, and using the fact that the problem lies in AC^0 for polylogarithmic thresholds [23] yields the fact that its parameterized version lies in para-AC^0. However, this result requires profound result of circuit complexity and is rather involved, but using color coding we can give a very simple proof of this fact:

▶ **Lemma 3.3.** *p-THRESHOLD $\in$ para-AC^0.*

3.2 Finding Embeddings of Graphs of Bounded Tree-Width and Depth

A different way of looking at the matching problem is to see it as an embedding problem: Instead of trying to find k edges in a graph G that have no endpoints in common, we can try to "embed" the graph $H = kK_2$, consisting of k isolated edges, into G. The advantage of this different point of view is, of course, that it generalizes nicely:

▶ **Problem 3.4** (p-EMB($\mathcal{H}$) for some class $\mathcal{H}$ of undirected graphs).

Instance: *Two undirected graphs $H = (V_H, E_H) \in \mathcal{H}$ and $G = (V_G, E_G)$.*
Parameter: *H*
Question: *Is there a injective homomorphism $\phi\colon V_H \to V_G$, that is, is H isomorphic to a (not necessarily induced) subgraph of G?*

For arbitrary $\mathcal{H}$, the problem is easily be seen to be W[1]-hard by a reduction from p-CLIQUE. However, for restricted $\mathcal{H}$, the problem becomes fixed-parameter tractable. The best results so far are by Chen and Müller [10] who show that when $\mathcal{H}$ has bounded tree-depth, p-EMB($\mathcal{H}$) $\in$ para-L; when $\mathcal{H}$ has bounded path-width, p-EMB($\mathcal{H}$) is the para-L-reduction closure of the distance problem in graphs, parameterized by the distance; and when $\mathcal{H}$ has bounded tree-width, p-EMB($\mathcal{H}$) is the para-L-reduction closure of the embedding problem for trees, parameterized by the tree-size. In contrast to these results, Amano showed for the unparameterized setting, in which we consider the size of H to be a constant, that the problem can be solved in AC^0 with similar techniques [3]. We improve considerably on the first result of Chen and Müller by proving that embeddings of graphs of bounded tree-depth can actually be computed in para-AC^0. We complement their other results, without improving them, by showing that for graphs of bounded tree-width (and, thereby, also for bounded path-width) the embedding problem lies in para-$\mathrm{AC}^{0\uparrow}$.

In order to formulate our results, we first need to review the definition of a tree-decomposition, see [17] for a more detailed introduction. A *tree-decomposition* of a graph $H = (V, E)$ is a tree T together with a mapping ι from the nodes of T to subsets (called *bags*) of V such that (1) for every edge $\{u, v\} \in E$ there is some bag containing u and v, that is, there is some $x \in V$ with $\{u, v\} \subseteq \iota(x)$ and (2) for every vertex $x \in V$ the set of nodes of T whose bags contain x forms a connected subset of T. The *width* of tree-decomposition is the size of its largest bag minus 1, its depth is the maximum of the width and the depth of T. Define $\mathrm{tw}(H)$ as the minimum width any tree-decomposition of H must have; define $\mathrm{td}(H)$ similarly for the tree-depth.

▶ **Theorem 3.5.** *Given two graphs $H = (V_H, E_H)$ and $G = (V_G, E_G)$ together with a tree-decomposition (T, ι) of H. An embedding of H into G can be computed by an* AC*-circuit of depth $O\big(\mathrm{depth}(T)\big)$ and size $f(|V_H|) \cdot O\big(|V_G|^{\mathrm{width}(T)}\big)$, if such an embedding exists.*

Sketch of Proof. Color the vertices of H uniquely and compute a $(|V_G|, |V_H|, |V_H|)$-universal coloring family. Starting from the leaves of the tree-decomposition, merge compatible partial

homomorphisms for the vertices of the bags until we reach the root of the decomposition, and, thus, obtain a homomorphism for H. The number of iterative steps required for this equals the depth of the tree-decomposition. ◀

If H is a parameter, we can compute a width- or depth-bounded tree-decomposition (T, ι) of H in a preprocessing step. This implies the following corollaries:

▶ **Corollary 3.6.** *Let $\mathcal{H}$ be the class of all graphs of tree-width at most d for some constant d. Then p-EMB$(\mathcal{H}) \in$ para-AC$_{f(|H|)} \subseteq$ para-AC$^{0\uparrow}$.*

▶ **Corollary 3.7.** *Let $\mathcal{H}$ be the class of all graphs of tree-depth at most d for some constant d. Then p-EMB$(\mathcal{H}) \in$ para-AC^0.*

We make two remarks at this point: First, one cannot generalize Theorem 3.5 to clique-width since the embedding problem for cliques, which have clique-width 1, is already hard for W[1]. Second, the theorem and the corollary also hold for relational structures $\mathcal{H}$ and $\mathcal{G}$ and if we bound the tree-width of $\mathcal{H}$'s Gaifman graph. Since paths have tree-width 1, the complexity of one of the canonical problems for color-coding – the p_k-PATH problem – can be determined: p_k-PATH $\in$ para-AC$^{0\uparrow}$. This allows us to give a short proof of the following lemma on the complexity of the distance problem for directed graphs where the distance is the parameter (one can also prove this lemma directly quite easily):

▶ **Lemma 3.8.** p_d-DISTANCE $\in$ para-AC$_{f(d)} \subseteq$ para-AC$^{0\uparrow}$.

A known fact from circuit complexity states that a polynomial-sized AC-circuit that decides whether a given graph G contains a path of length at most d between to vertices s and t requires depth $\Omega(\log \log d)$ [5]. This implies p_d-DISTANCE $\notin$ para-AC^0.

▶ **Corollary 3.9.** para-AC$^0 \subsetneq$ para-AC$^{0\uparrow}$.

3.3 First-Order Model Checking

Our last result in this section on tools is an algorithmic meta-theorem: We show that the model checking problem for first-order logic on graphs of bounded degree lies in para-AC$^{0\uparrow}$. We build strongly on a previous result by Flum and Grohe [16], who showed that this model checking problem lies in para-L, but differ in three regards: First, we use color coding in our proof, which simplifies the argument somewhat, second, we identify the parameterized distance problem on bounded degree graphs as the only part of the computation that is presumably not in para-AC^0, and, third, we observe that the degree of the graphs can be made a parameter and need not be considered constant.

▶ **Problem 3.10** ($p_{\phi,\delta}$-MC(FO))**.**
Instance: *A logical structure $\mathcal{A}$ and a first-order formula ϕ.*
Parameter: *The (size of) the formula ϕ and the maximum degree δ of $\mathcal{A}$'s Gaifman graph.*
Question: *$\mathcal{A} \models \phi$?*

▶ **Theorem 3.11.** $p_{\phi,\delta}$-MC(FO) $\in$ para-AC$_{f(\phi+\delta)} \subseteq$ para-AC$^{0\uparrow}$.

Sketch of Proof. By Gaifman's Theorem [20] we can rewrite the given formula as a formula ϕ' in Gaifman normal form. Thus, what essentially remains is to check whether the structure (which we can interpret as a graph) contains k disjoint "balls" of size bounded in the parameter (due to the maximum degree of the underlying Gaifman graph) that satisfy the subformulas in ϕ'. To find these substructures, we make use of color coding and apply Lemma 3.8 to compute the corresponding connecting components. Finally, we only have to model check the resulting parameter-sized substructures. ◀

We conclude with the remark that the depth of the circuits constructed in the above theorem just depends on the degree δ of the graph and on the radius r of the balls, which measure how "local" the formula ϕ is. The smallest r for which ϕ can be rewritten as in the proof is known as the *locality rank* $\mathrm{lr}(\phi)$ and the proof actually shows that $p_{\phi,\delta}$-MC(FO) $\in$ para-$\mathrm{AC}_{O(\delta^{\mathrm{lr}(\phi)})}$.

4 Fast Parallel Fixed-Parameter Algorithms for Natural Problems

The tools we have developed are now applied to a number of natural parameterized problems found in the literature.

Packing Problems. We have already pointed out that the parameterized matching problem can be seen as an embedding problem, where the objective is to embed the graph $H = kK_2$, consisting of k disjoint copies of a single edge, into a graph G. Embedding multiple disjoint copies of the same graph into another graph is also known as "packing". Clearly, instead of edges we can also pack other things as long as taking any number of copies of these "other things" still has bounded tree-depth. For instance, we can try to "pack" k different triangles into G, that is, we can check whether there are k vertex-disjoint triangles in G. Unlike the matching problem, triangle packing is known to be NP-complete.

▶ **Theorem 4.1.** p-TRIANGLE-PACKING $\in$ para-AC^0.

Proof. Just observe that a graph H consisting of any number of disjoint copies of a triangle has tree-depth 3. The claim follows from Corollary 3.7. ◀

Indeed, for any fixed graph H_0 the packing problem p-H_0-PACKING lies in para-AC^0, where the question is whether we can find k disjoint copies of H_0 in G and k is the parameter:

▶ **Theorem 4.2.** p-H_0-PACKING $\in$ para-AC^0 *for every fixed graph* H_0.

Further variants arise when, instead of a single graph H_0, we are given a whole multiset of graphs as inputs and we must find disjoint copies of all of them in G. Again, as long as there is a fixed bound on the size of the graphs, the tree-depth of their disjoint union is bounded and, hence, the packing problem lies in para-AC^0.

The complexity of packing problem changes when the to-be-packed graphs no longer have constant size as in the following problem:

▶ **Problem 4.3** ($p_{k,l}$-CYCLE-PACKING)**.**
Instance: *An undirected graph G and two numbers k and l.*
Parameter: *k and l*
Question: *Are there k vertex-disjoint cycles in G, each having length l?*

The graph $H = kC_l$ consisting of k copies of a cycle of length l no longer has bounded tree-depth; it does have tree-width 2, however. Thus, by Theorem 3.5 we get:

▶ **Theorem 4.4.** $p_{k,l}$-CYCLE-PACKING $\in$ para-$\mathrm{AC}_{f(k+l)} \subseteq$ para-$\mathrm{AC}^{0\uparrow}$.

The same result obviously also holds for $p_{k,l}$-PATH-PACKING and it also holds for p-FOREST-PACKING, where we are given a forest as input and the parameter is the total numbers of vertices in it. We conclude with the remark that these ideas cannot be extended to packing graphs whose tree-width is not bounded: Already embedding cliques, let alone packing them, is W[1]-hard.

Covering Problems. In *covering problems* we must choose vertices in a graph (or sometimes hypergraph) such that all (p-VERTEX-COVER) or some (p-PARTIAL-VERTEX-COVER) of the edges are "covered," that is, they intersect with the set of chosen vertices. The best-known covering problem is undoubtedly p-VERTEX-COVER, whose complexity has been scrutinized extensively in parameterized complexity theory. We now prove p-VERTEX-COVER $\in$ para-AC^0; a fact that nicely reflects on a theoretical basis the "empirical" observation that p-VERTEX-COVER is one of the "easiest" parameterized problems. The problem was one of the first shown to lie in para-P, was then shown to lie in para-L by Cai et al. [8], then in para-TC^0 by Elberfeld and the last two authors [14].

▶ **Theorem 4.5.** *p-VERTEX-COVER $\in$ para-AC^0.*

Partial covering problems ask us not to cover all edges, but only t of them:

▶ **Problem 4.6** ($p_{k,t}$-PARTIAL-VERTEX-COVER)**.**

Instance: *An undirected graph $G = (V, E)$ and two numbers k and t.*
Parameter: *k, t*
Question: *Is there a set $S \subseteq V$ of cardinality $|S|$ at most k such that the cardinality of $\big\{\{u, v\} \in E \mid u \in S \vee v \in S\big\}$ is at least t?*

Another version is p_t-EXACT-PARTIAL-VERTEX-COVER, where the size of S is no longer restricted, but the cardinality of $\big\{\{u, v\} \in E \mid u \in S \vee v \in S\big\}$ must be *exactly* t.

These problems, which are generally considered to be harder than the plain vertex cover problem, lie in the class para-AC$^{0\uparrow}$. Our proofs make an interesting use of Theorem 3.11. Recall that this "meta-theorem" states that all first-order properties of graphs, parameterized by the first-order property and the maximum degree of the graph, can be decided in para-AC$^{0\uparrow}$. Covering properties can be expressed using first-order formulas – but we make *no* requirement concerning the degree of the input graph. The trick is to first reduce the inputs to graphs of bounded degree and then apply the meta theorem. Such a two-step approach is typically in advanced applications of algorithmic meta-theorems.

▶ **Theorem 4.7.** *$p_{k,t}$-PARTIAL-VERTEX-COVER $\in$ para-AC$_{f(k+t)} \subseteq$ para-AC$^{0\uparrow}$.*

▶ **Theorem 4.8.** *p_t-EXACT-PARTIAL-VERTEX-COVER $\in$ para-AC$_{f(t)} \subseteq$ para-AC$^{0\uparrow}$.*

We conclude with the remark that the above results on finding vertex coverings for graphs cannot easily be extended to hypergraphs since for hypergraphs covering problems are typically hard for at least W[1].

Clustering Problems. Clustering algorithms have a wide variety of applications, for example in computational biology where we want to cluster genes and proteins or process transcription data [7]. A basic clustering problem for graphs is the following:

▶ **Problem 4.9** ($p_{k,\ell}$-CLUSTER-EDITING)**.**

Instance: *An undirected graph $G = (V, E)$ and a numbers ℓ and k.*
Parameter: *ℓ, k*
Question: *Can we add and/or delete up to k edges to or from G such that the resulting graph consists of ℓ connected components, each of which is a clique?*

A variant is p_k-MANY-CLUSTER-EDITING, where we just require that the edited graph consists of cliques and do not prescribe the number of clusters beforehand. This variant has been extensively studied, most notably by Gramm et al. [21] and Böcker [6] who showed its

fixed-parameter tractability. For the first version, algorithms based on color coding result in reasonable running times, but where recently be outperformed by other approaches [19]. However, using a color coding approach is useful when we consider parallel algorithms:

▶ **Theorem 4.10.** $p_{k,\ell}$-CLUSTER-EDITING $\in$ para-AC^0.

▶ **Corollary 4.11.** p_k-MANY-CLUSTER-EDITING $\in$ para-AC^0.

We remark that if ℓ is not no longer considered a parameter in cluster editing, the problem complexity increases only moderately:

▶ **Corollary 4.12.** p_k-CLUSTER-EDITING $\in$ para-TC^0.

Theorem 4.10 has another interesting corollary: Let $p_{k,p}$-COMPLETE-p-PARTITE-EDITING be the problem of determining whether in a graph G we can add and / or remove up to k edges such that the resulting graph is complete p-partite, that is, its vertex set can be partitioned into exactly p non-empty sets such that there is an edge between two vertices if, and only if, they belong to two different sets. Since the complement of a complete p-partite graph is exactly a collection of p cliques, we get the following corollary:

▶ **Corollary 4.13.**
1. $p_{k,p}$-COMPLETE-p-PARTITE-EDITING $\in$ para-AC^0.
2. p_k-COMPLETE-p-PARTITE-EDITING $\in$ para-TC^0.

Finally, instead of looking for just one complete p-partite graph, we can look for several at the same time:

▶ **Problem 4.14** ($p_{k,p}$-MULTIPARTITE-CLUSTER-EDITING).

Instance: *An undirected graph $G = (V, E)$, a natural number k, and a sequence of natural numbers $p_1, p_2, \ldots, p_\ell$.*
Parameter: *k, $p = p_1 + \cdots + p_\ell$*
Question: *Can we add or delete k edges of G such that the resulting graph consist of d connected components C_1 to C_ℓ such that each C_i is a complete p_i-partite graph?*

▶ **Theorem 4.15.**
1. $p_{k,p}$-MULTIPARTITE-CLUSTER-EDITING $\in$ para-AC^0
2. $p_{k,\ell}$-MULTIPARTITE-CLUSTER-EDITING $\in$ para-TC^0.

Graph Separation Problems. Graph separation problems are problems where we ask to separate a set of ℓ vertices from the remaining graph by deleting at most k other vertices. They play a key role in many real-world network applications like finding communities or isolating dangerous vertices. While this problem is well-known to be NP-complete in the unparameterized setting and W[1]-hard in the parameterized setting for parameters k, ℓ, and $k + \ell$, the complexity of the problem changes dramatically if we require the separated set of vertices to be connected:

▶ **Problem 4.16** ($p_{k,\ell}$-CUTTING-ℓ-CONNECTED-VERTICES).

Instance: *An undirected graph $G = (V, E)$ and two natural numbers k and ℓ.*
Parameter: *k, ℓ*
Question: *Is there a partitioning of V into three sets X, S, and Y with $|X| = \ell$ and $|S| \leq k$ such that X is connected and for all $\{x, y\} \in E$ with $x \in X$ we have $y \notin Y$?*

Marx [22] showed that this problem is fixed-parameter tractable; Fomin, Golovach, and Korhonen [18] studied a similar version, namely $p_{k,\ell}$-CUTTING-AT-MOST-ℓ-VERTICES, in which the set X is not required to be connected and may be of size *at most* ℓ, i. e., $1 < |X| \leq \ell$, and for which Fomin et al. gave an FPT-algorithm based on color coding. The main idea is to colorize the given graph with two colors such that the vertices of the set X get colored with the first color and the vertices in S get the second color. Thus, we only have to find the solution within the vertices of the first color. This algorithm can be implemented in para-$\mathrm{AC}^{0\uparrow}$ and, moreover, works for $p_{k,\ell}$-CUTTING-ℓ-CONNECTED-VERTICES as well.

▶ **Theorem 4.17.**

1. $p_{k,\ell}$-CUTTING-ℓ-CONNECTED-VERTICES $\in$ para-$\mathrm{AC}_{f(\ell)} \subseteq$ para-$\mathrm{AC}^{0\uparrow}$.
2. $p_{k,\ell}$-CUTTING-AT-MOST-ℓ-VERTICES $\in$ para-$\mathrm{AC}_{f(\ell)} \subseteq$ para-$\mathrm{AC}^{0\uparrow}$.

We conclude with the remark that both problems can also be solved with algorithms similar to the ones presented above if we consider the *terminal versions* of these problems [18], i. e., there is a special terminal vertex t which has to be part of X. For this, we have to modify the above algorithms to consider only blue components that contain t.

5 Conclusion

We have seen that many natural parameterized problems can be solved in constant parallel time or in parallel time depending only on the parameters while doing only "FPT work." We stress that our results are of a theoretical nature and do not directly give practical parallel implementations for the problems presented; but they show that such implementations are possible *in principle* for them. The core technique used in all proofs (at least indirectly) was *color coding,* which can be done in constant time and which is already used in practice.

This paper did not address lower bounds. While for para-AC^0 this is not problematic since this class lies at the bottom of almost any hierarchy of parameterized classes, some problems in para-$\mathrm{AC}^{0\uparrow}$ might well "fall down" to para-AC^0. Here we only know a explicit lower bound for the distance problem, which does not lie in para-AC^0. Establishing lower bounds for other problems in para-$\mathrm{AC}^{0\uparrow}$ is therefore a reasonable research goal.

References

1 F. N. Abu-Khzam, M. A. Langston, P. Shanbhag, and C. T. Symons. Scalable Parallel Algorithms for FPT Problems. *Algorithmica*, 45(3):269–284, 2006.

2 N. Alon, R. Yuster, and U. Zwick. Color-Coding. *Journal of the ACM*, 42(4):844–856, 1995.

3 Kazuyuki Amano. k-Subgraph Isomorphism on AC^0 Circuits. *Computational Complexity*, 19(2):1016–3328, 2010.

4 David A. Mix Barrington and Neil Immerman. On uniformity within NC^1. *Journal of Computer and System Sciences*, 41(3):274–306, 1990.

5 Paul Beame, Russell Impagliazzo, and Toniann Pitassi. Improved depth lower bounds for small distance connectivity. *Computational Complexity*, 7(4):325–345, 1998.

6 S. Böcker. A Golden Ratio Parameterized Algorithm for Cluster Editing. In *Proceedings of the Twenty-Second International Workshop on Combinatorial Algorithms*, volume 7056 of *IWOCA'11*, pages 85–95. Springer Berlin Heidelberg, 2011.

7 S. Böcker and J. Baumbach. Cluster Editing. In *Proceedings of the Ninth Conference on Computability in Europe*, volume 7921 of *CiE'13*, pages 33–44. Springer Berlin Heidelberg, 2013.

8 L. Cai, J. Chen, R. G. Downey, and M. R. Fellows. Advice Classes of Parameterized Tractability. *Annals of Pure and Applied Logic*, 84(1):119–138, 1997.

9 Marco Cesati and Miriam Di Ianni. Parameterized parallel complexity. In *Proceedings of the Fourth International Euro-Par Conference*, volume 1470 of *Euro-Par'89*, pages 892–896. Springer Berlin Heidelberg, 1998.

10 H. Chen and M. Müller. The Fine Classification of Conjunctive Queries and Parameterized Logarithmic Space. *ACM Transactions on Computation Theory*, 7(2):7:1–7:27, 2014.

11 Y. Chen, J. Flum, and M. Grohe. Bounded Nondeterminism and Alternation in Parameterized Complexity Theory. In *Proceedings of the Eighteenth IEEE Conference on Computational Complexity*, CCC'03, pages 13–29. IEEE Computer Society, Los Alamitos, California, 2003.

12 R.G. Downey and M.R. Fellows. *Parameterized Complexity*. Springer, 1999.

13 Rodney G. Downey, Michael R. Fellows, and Kenneth W. Regan. Parameterized Circuit Complexity and the W Hierarchy. *Theoretical Computer Science*, 191(1–2):97–115, 1998.

14 M. Elberfeld, C. Stockhusen, and T. Tantau. On the Space Complexity of Parameterized Problems: Classes and Completness. *Algorithmica*, 71(3):661–701, 2014.

15 M. Fellows, P. Heggernes, F. Rosamond, C. Sloper, and J. A. Telle. Finding k Disjoint Triangles in an Arbitrary Graph. In *Proceedings of the Thirtieth International Workshop on Graph-Theoretic Concepts in Computer Science*, volume 3353 of *WG'04*, pages 235–244. Springer Berlin Heidelberg, 2004.

16 J. Flum and M. Grohe. Describing Parameterized Complexity Classes. In *Proceedings of the Nineteenth Annual Symposium on Theoretical Aspects of Computer Science*, volume 2285 of *STACS'02*, pages 359–371. Springer Berlin Heidelberg, 2002.

17 J. Flum and M. Grohe. *Parameterized Complexity Theory*. Springer, 2006.

18 F. V. Fomin, P. A. Golovach, and J. H. Korhonen. On the Parameterized Complexity of Cutting a Few Vertices from a Graph. In *Proceedings of the Thirty-Eight International Symposium of Mathematical Foundations of Computer Science*, volume 8087 of *MFCS '13*, pages 421–432. Springer, Heidelberg, Germany, 2013.

19 F. V. Fomin, S. Kratsch, M. Pilipczuk, M. Pilipczuk, and Y. Villanger. Tight Bounds for Parameterized Complexity of Cluster Editing. In *Proceedings of the Thirtieth International Symposium on Theoretical Aspects of Computer Science*, volume 20 of *STACS'13*, pages 30–43. International Symposium on Theoretical Aspects of Computer Science, 2013.

20 H. Gaifman. On Local and Non-Local Properties. In *Proceedings of the Herbrand Symposium, Logic Colloquium 1981*, pages 105–135. North Holland, 1982.

21 J. Gramm, J. Guo, F. Hüffner, and R. Niedermeier. Graph-Modeled Data Clustering: Fixed-Parameter Algorithms for Clique Generation. In *Proceedings of the Fifth Italian Conference of Algorithms and Complexity*, volume 2653 of *CIAC'03*, pages 108–119. Springer Berlin Heidelberg, 2003.

22 D. Marx. Parameterized Graph Separation Problems. *Theoretical Computer Science*, 351(3):394–406, 2006.

23 I. Newman, P. Ragde, and A. Wigderson. Perfect Hashing, Graph Entropy, and Circuit Complexity. In *Proceedings of the Fifth Annual Structure in Complexity Theory Conference*, pages 91–99. IEEE Computer Society, Los Alamitos, California, 1990.

24 H. Vollmer. *Introduction to Circuit Complexity*. Springer, 1999.