

On the Parallel Parameterized Complexity of MaxSAT Variants

Max Bannach ✉ 

Institute for Theoretical Computer Science, Universität zu Lübeck, Germany

Malte Skambath ✉ 

Department of Computer Science, Universität Kiel, Germany

Till Tantau ✉

Institute for Theoretical Computer Science, Universität zu Lübeck, Germany

Abstract

In the maximum satisfiability problem (MAX-SAT) we are given a propositional formula in conjunctive normal form and have to find an assignment that satisfies as many clauses as possible. We study the *parallel* parameterized complexity of various versions of MAX-SAT and provide the first *constant-time* algorithms parameterized either by the solution size or by the allowed excess relative to some guarantee (“above guarantee” versions). For the *dual parameterized* version where the parameter is the number of clauses we are allowed to leave unsatisfied, we present the first parallel algorithm for MAX-2SAT (known as ALMOST-2SAT). The difficulty in solving ALMOST-2SAT in *parallel* comes from the fact that the iterative compression method, originally developed to prove that the problem is fixed-parameter tractable at all, is inherently *sequential*. We observe that a graph flow whose value is a parameter can be computed in parallel and use this fact to develop a parallel algorithm for the vertex cover problem parameterized above the size of a given matching. Finally, we study the parallel complexity of MAX-SAT parameterized by the vertex cover number, the treedepth, the feedback vertex set number, and the treewidth of the input’s incidence graph. While MAX-SAT is fixed-parameter tractable for all of these parameters, we show that they allow different degrees of possible parallelization. For all four we develop dedicated parallel algorithms that are *constructive*, meaning that they output an optimal assignment – in contrast to results that can be obtained by parallel meta-theorems, which often only solve the decision version.

2012 ACM Subject Classification Theory of computation → Parallel computing models; Theory of computation → Fixed parameter tractability

Keywords and phrases max-sat, almost-sat, parallel algorithms, fixed-parameter tractability

Digital Object Identifier 10.4230/LIPIcs.SAT.2022.19

Related Version *Full Version*: <https://arxiv.org/abs/2206.01280>

1 Introduction

Maximum satisfiability problems ask us to find solutions for constraint systems that satisfy as many constraints as possible. The perhaps best-studied version is MAX-SAT, where the constraint system is a propositional formula in conjunctive normal form, and the goal is to find an assignment that satisfies the largest number of clauses possible. The problem is NP-complete even restricted to formulas with at most two literals per clause [23]. It is also the canonical complete problem for the optimization class MaxSNP and, thus, a central topic in the research of approximation algorithms [38]. Many real-world problems can be encoded as MAX-SAT instances, which led to the successful development of exact solvers (see Chapter 23 and 24 in [7]). Following the positive example of SAT solvers, these tools became ever better over the last decades – regularly breaking alleged theoretical barriers in practice. In search of an explanation for this phenomenon, theoreticians studied the parameterized complexity of MAX-SAT [2, 14, 18, 31, 35, 41], which resulted in new concepts such as *parameterization above a guarantee* [34] or *dual parameterizations* [40].



© Max Bannach, Malte Skambath, and Till Tantau;
licensed under Creative Commons License CC-BY 4.0

25th International Conference on Theory and Applications of Satisfiability Testing (SAT 2022).

Editors: Kuldeep S. Meel and Ofer Strichman; Article No. 19; pp. 19:1–19:19

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

With membership in the class para-P (or FPT) of fixed-parameter tractable problems settled for many variants of MAX-SAT, a new question has surfaced both in theoretical and practical research over the last decade: Which problems admit *parallel* fpt-algorithms, i. e., which problems lie in para-NC, the parameterized version of NC? The vertex cover problem is the poster child for such a problem as it lies even in para-AC⁰, which is the smallest commonly studied parameterized class and can be thought of as “solvable with fpt-many parallel processing units in constant time” [3]. Many of the important tools underlying fpt-theory, such as search trees, graph decompositions, or kernelizations, have been adapted to the parallel setting by different research groups [1, 3, 10, 39].

In this paper we study the parallel complexity of maximum satisfiability problems for various parameterizations. We show that the parallel fpt-toolkit can be used to establish parallel algorithms for MAX-SAT parameterized by the solution size or parameterized above some guarantee. We also develop dedicated algorithms for the problem parameterized by the structural parameters *treewidth*, *feedback vertex set number*, *treedepth*, and *vertex cover number* and observe an ever higher level of achievable parallelization. Our most technical contribution is a parallel algorithm for p_k -ALMOST-2SAT, which is MAX-2SAT for the dual parameterization where we try to satisfy at least $m - k$ clauses in a given 2CNF formula (m is always the number of clauses, n the number of variables, k a positive integer parameter, and “p-” indicates a parameterized problem with the index being the parameter; variables occurring in problem names such as in *dSAT* are fixed constants). This problem has stubbornly resisted all known techniques in the parallel fpt-toolkit: First, one cannot use algorithmic meta-theorems that are often used to show membership in para-NC. The algorithmic meta-theorems for second-order logic [4] fail as the underlying incidence graphs generally do not have bounded treewidth, and those for first-order logic [11, 21, 39] fail as the satisfiability of a 2CNF formula is not first-order definable. Second, the central tool for showing that it lies in para-P, namely iterative compression [40, 41], is – as the name suggests – highly *sequential*.

We develop new tools that go beyond the established toolkits and involve two ideas. First, we make a simple, but non-trivial, observation concerning the parallel computation of graph flows. While computing graph flows is P-complete [27] and, thus, most likely *not* parallelizable and while even computing a 0-1-flow in parallel is a long standing open problem [33], we observe that computing a flow of *parameter value* k can be done in k consecutive rounds of a *parallel* Ford-Fulkerson [22] step. The second idea is more complex, as we study a seemingly different problem: vertex cover, but not with the sought size of the vertex cover as the parameter, but with the (smaller and hence less restrictive) parameter “integrality excess of the LP.” An fpt-reduction from parameterized ALMOST-2SAT to this vertex cover version is well known [40]. To compute vertex covers for this “looser” parameter in parallel, we combine results by Iwata, Oka and Yoshida [31] on the properties of the Hochbaum network underlying the linear program and apply the earlier-mentioned observations on graph flows.

Contribution I. We settle the parallel complexity of MAX-SAT for the canonical parameters k (solution size) and g (solution size minus $\lceil \frac{m}{2} \rceil$): p_k -MAX-SAT \in para-AC⁰, but p_g -MAX-SAT is para-TC⁰-complete. If we assume that clauses have size exactly d (MAX-EdSAT), we show that an “above average version” lies in para-AC⁰ as well – a version that is known to be para-NP-hard if the size of the clauses is unbounded.

Contribution II. We study variants of p_k -ALMOST-SAT, i. e., of MAX-SAT parameterized dually, and present, for the first time, parallel algorithms for this problem on various classes of CNFs. The main achievement is a para-NC algorithm for the problem restricted to 2CNFs.

Contribution III. The structural parameters *vertex cover number*, *treedepth*, *feedback vertex set number*, and *treewidth* are partially ordered, meaning that graphs of bounded vertex cover number have bounded treedepth and so on. It is known that MAX-SAT is in para-P parameterized by any of these, but the sequential algorithms tend to hide beneficial properties gained by more restrictive parameterizations. We show that we obtain a higher level of parallelization for larger parameters (reaching from para-TC⁰ and para-TC⁰↑, over para-TC¹↑, up to para-AC²↑). Additionally, our algorithms are constructive (they output an optimal assignment), which is in contrast to existing parallel meta-theorems.

Table 1 provides an overview of all results presented within this manuscript.

■ **Table 1** Variations of the maximum satisfiability problem studied within this paper. The lower bounds are the trivial ones, while the upper bounds are proven in the referenced theorems or lemmas. If the result is marked as constructive, a corresponding optimal assignment can be produced (this either is proven directly, or the presented algorithm can be modified in an obvious way). The blue headlines indicate the technique used to obtain the results.

Problem	Complexity Bound		Clause Size	Construct.	Reference
	Lower	Upper			
<i>Can Be Solved Using Color Coding</i>					
p _{k,t} -MAX-δ-CIRCUIT-SAT	para-AC ⁰	para-AC ⁰	unbounded	✓	Theorem 3.2
p _k -MAX-SAT	para-AC ⁰	para-AC ⁰	unbounded	✓	Corollary 3.3
p _k -MAX-NAE-SAT	para-AC ⁰	para-AC ⁰	unbounded	✓	Corollary 3.3
p _{k,d,x} -MAX-EXACT-SAT	para-AC ⁰	para-AC ⁰	≤ d	✓	Corollary 3.3
p _{k,d} -MAX-DNF	para-AC ⁰	para-AC ⁰	≤ d	✓	Corollary 3.3
p _g -MAX-SAT-ABOVE-HALF	para-TC ⁰	para-TC ⁰	unbounded	✗	Theorem 3.5
<i>Can Be Solved Using Algebraic Techniques</i>					
p _g -MAX-EdSAT-ABOVE-AVERAGE	para-AC ⁰	para-AC ⁰	= d	✗	Lemma 3.7
<i>Can Be Solved Using Graph Flows</i>					
p _k -ALMOST-NAE-2SAT	para-L	para-NL [↑]	≤ 2	✓	Theorem 4.2
p _k -ALMOST-2SAT	para-NL	para-NL [↑]	≤ 2	✓	Theorem 4.2
<i>Can Be Solved Using Graph Extensions</i>					
p _k -ALMOST-NAE-SAT(2)	para-L	para-L	unbounded	✗	Theorem 4.24
p _k -ALMOST-SAT(2)	para-L	para-L	unbounded	✗	Theorem 4.24
<i>Can Be Solved Using Reduction to Vertex Cover</i>					
p _k -ALMOST-DNF	para-AC ⁰	para-AC ⁰	unbounded	✓	Theorem 4.27
p _k -MIN-SAT	para-AC ⁰	para-AC ⁰	unbounded	✓	Lemma 4.28
<i>Can Be Solved Using Dynamic Programming</i>					
p _{vc} -PARTIAL-MAX-SAT	para-TC ⁰	para-TC ⁰	unbounded	✓	Theorem 5.1
p _{td} -PARTIAL-MAX-SAT	para-TC ⁰	para-TC ⁰ ↑	unbounded	✓	Theorem 5.1
p _{fvS} -PARTIAL-MAX-SAT	para-L	para-TC ¹ ↑	unbounded	✓	Theorem 5.1
p _{tw} -PARTIAL-MAX-SAT	para-L	para-AC ² ↑	unbounded	✓	Theorem 5.1

As byproducts, we establish results that may be of independent interest: First, we present an alternative characterization of the “up-classes”. Second, we lower the complexity of the feedback vertex set problem to para-L[↑], which is obtained “by iterating a para-L computation parameter-many times.” Third, we obtain para-NC algorithms for problems that can be reduced to p_k-ALMOST-2SAT which includes, in particular, the odd cycle transversal problem (can we make a given graph bipartite by deleting *k* vertices?).

Related Work. The parameterized complexity of MAX-SAT is an active field of research dating back the pioneering work by Mahajan and Raman [34]. Since then, parameterized algorithms for ever looser parameters have been found [13, 15, 29] or their existence has been refuted [14]. This research has also branched out into the study of preprocessing algorithms [25, 26], parameterized heuristics [42], and algorithms utilizing structural decompositions [18, 28]. However, to the best of our knowledge, not yet to parallel parameterized algorithms.

While research on parallel fixed-parameter algorithms dates back to the early 1990s to the study of the *space* complexity of parameterized problem [8] (via the inclusion chain $\text{NC}^1 \subseteq \text{L} \subseteq \text{NL} \subseteq \text{AC}^1$), a systematic study of parallel fixed-parameter algorithms started only in the last decade [19]. Since then, a toolbox has been compiled that contains algorithmic meta-theorems both for monadic second-order logic [4] and for first-order logic [11, 21, 39].

Organization of this Paper. After some preliminaries in the next section, we study MAX-SAT parameterized by the solution size and parameterized above a guarantee in Section 3. We continue and study MAX-SAT variants with a dual parameterization in Section 4. The largest and technical most involved part here is a parallel algorithm for p_k -ALMOST-2SAT. Finally, we consider structural parameterizations of MAX-SAT in Section 5 and establish a connection between the level of parallelization we can achieve and the used parameter.

2 Background on Parameterized Problems and Classes

Propositional Logic and MaxSAT. We assume an infinite supply of *propositional variables* x_1, x_2, \dots and call a variable x or its negation $\neg x$ a *literal*. A *propositional formula in conjunctive normal form* (a CNF) ϕ is a conjunction of disjunctions of literals, for instance $\phi = (x_1 \vee x_2 \vee \neg x_2) \wedge (x_1) \wedge (x_1) \wedge (x_2 \vee x_2)$. We write $\text{vars}(\phi)$ for the set of variables in ϕ and $\text{clauses}(\phi)$ for the multiset of clauses, which are the sets of literals in the disjunctions, e. g., $\text{clauses}(\phi) = \{\{x_1, x_2, \neg x_2\}, \{x_1\}, \{x_1\}, \{x_2\}\}$. We denote $|\text{vars}(\phi)|$ by n and $|\text{clauses}(\phi)|$ by m (so $n = 2$ and $m = 4$ in the example), and let m_\emptyset be the number of empty clauses.

An *assignment* $\beta: \text{vars}(\phi) \rightarrow \{0, 1\}$ maps every variable of ϕ to a truth value. It satisfies a literal ℓ if $\ell = x$ and $\beta(x) = 1$ or if $\ell = \neg x$ and $\beta(x) = 0$. Furthermore, it *satisfies* a clause C (denoted by $\beta \models C$) if it satisfies at least one literal in it; it *nae-satisfies* a clause if it additionally falsifies at least one literal (“not-all-equal-satisfies”).

The MAX-SAT problem asks, given a CNF ϕ and a number k , whether there is an assignment β that satisfies at least k clauses. If β satisfies all m clauses, then $\beta \models \phi$, i. e., β is a model of ϕ . Variations are obtained by modifying the condition of a clause being satisfied, e. g., in MAX-NAE-SAT we seek an assignment that nae-satisfies at least k clauses.

Graphs, Networks, and Flows. In this paper, graphs are pairs $G = (V, E)$ of finite sets of vertices and edges. In this context, n denotes $|V|$ and m denotes $|E|$. For undirected graphs, edges are two-element subsets of V , for directed graphs (digraphs) $E \subseteq V \times V$. A *walk in G of length p* is a sequence (v_0, \dots, v_p) of vertices $v_i \in V$ with $(v_i, v_{i+1}) \in E$ (or $\{v_i, v_{i+1}\} \in E$ for undirected graphs) for all $i \in \{0, \dots, p-1\}$. A *path* is a walk in which all vertices (and hence all edges) are distinct. A *cycle* is a walk of length at least 3 in which all vertices are distinct except for the first and last, which must be identical. For a set $S \subseteq V$ we write $G - S$ for the graph induced on the set $V \setminus S$. For an undirected graph G the *neighborhood* $N(v)$ of a vertex v is the set $\{u \in V \mid \{u, v\} \in E\}$, the *degree* of v is $|N(v)|$.

We think of digraphs $G = (V, E)$ with two designated vertices $s, t \in V$ as *networks*, and we always assume that in networks between any two different vertices u and v at most one edge is present (either (u, v) or (v, u)) – if this is not the case we may simply subdivide each edge.

A 0-1-flow from s to t in G is a mapping $f: E \rightarrow \{0, 1\}$ such that for all $v \in V \setminus \{s, t\}$ we have $\sum_{(u,v) \in E} f(u, v) = \sum_{(v,w) \in E} f(v, w)$. The *value* $|f|$ of a flow is defined as the amount $|f| = \sum_{(s,v) \in E} f(s, v) - \sum_{(w,s) \in E} f(w, s)$ of flow leaving the source (or, equivalently, arriving at the target). For a flow f in a network G , the *residual graph* $R_f = (V, E_f)$ contains all edges of G that are not part of the flow and all reversed edges of the flow:

$$E_f = \{(u, v) \in E \mid f(u, v) = 0\} \cup \{(v, u) \in V \times V \mid f(u, v) = 1\}.$$

Standard Parameterized Problems and Complexity Classes. A *parameterized problem* is a set $Q \subseteq \Sigma^* \times \mathbb{N}$. In an instance (w, k) we call w the *input* (typically a CNF in this paper) and k the *parameter*. For instance, $\text{p}_k\text{-MAX-SAT} = \{(\phi, k) \mid \phi \text{ has an assignment satisfying at least } k \text{ clauses}\}$. We indicate the parameter as a subscript to the leading “p”.

A *parameterized function* is a mapping $F: \Sigma^* \times \mathbb{N} \rightarrow \Sigma^* \times \mathbb{N}$ such that the output parameter is bounded in terms of the input parameter, i. e., there is a function $b: \mathbb{N} \rightarrow \mathbb{N}$ with $k' \leq b(k)$ whenever $F(w, k) = (w', k')$. The *characteristic function* χ_Q of a parameterized problem Q maps $(w, k) \in Q$ to $(1, 0)$ and $(w, k) \notin Q$ to $(0, 0)$.

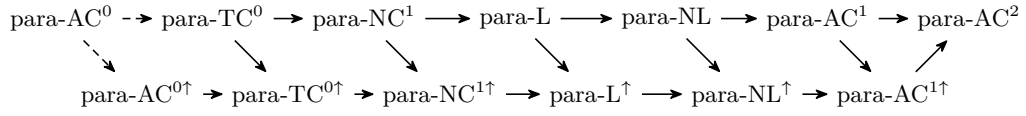
In parameterized complexity theory, the class para-P (also known as FPT) takes the role of P in classical complexity theory. A parameterized problem Q is in para-P if there is an algorithm that decides whether $(w, k) \in Q$ holds in time $f(k) \cdot n^{O(1)}$ for some computable function f . A *parallel* parameterized algorithm is able to decide the same question by a logarithmic-time-uniform¹ family of unbounded fan-in circuits of depth $O(\log^i n)$ for some fixed i (note that the depth does not depend on k) and size $f(k) \cdot n^{O(1)}$. The problem is then in the class para-AC^{*i*} or, in the presence of *threshold gates*, para-TC^{*i*}. Define para-NC as the union of all these para-AC^{*i*} classes or, equivalently, the union of all para-TC^{*i*} classes.

Up-Classes. The “up-arrow notation” was originally introduced in the context of parameterized circuit classes [3] to denote circuits that arise from taking a circuit of a certain depth (like $\log n$) and then allow “parameter-dependent-many layers” of such circuits (resulting in a depth like $f(k) \log n$). In this paper, we define the notation as the “closure of a parameterized function class under parameter-dependent-many iterations of linear functions,” which yields the same circuit classes, but also yields natural “up-versions” of para-L and para-NL. In detail, we take a parameterized function class and allow the functions in it to be applied to an input not just once, but rather “parameter-dependent-many times.” One must be a bit careful, though, to ensure that the intermediate results do not get too large. We require that the function we apply iteratively causes only a linear increase in the output size. For this, let us call a parameterized function F *linear* if $|F(w, k)| \leq f(k) \cdot |w|$ for some computable f .

► **Definition 2.1.** *Let para-FC be a class of parameterized functions. A parameterized function F lies in para-FC^\uparrow if there are (1) an “initial” function $I \in \text{para-FC}$, (2) a linear “iterator” function $L \in \text{para-FC}$, and (3) a computable “iteration number” function $r: \mathbb{N} \rightarrow \mathbb{N}$, such that $F(w, k) = L^{r(k)}(I(w, k))$, where L^r is the r -fold composition (or iteration) of L with itself. A problem lies in para-C^\uparrow if its characteristic function lies in para-FC^\uparrow .*

The following lemma shows that the definition is a generalization of the original definition of $\text{para-AC}^{i\uparrow}$ as the class of problems decidable by circuits of depth $f(k) \cdot O(\log^i n)$ and size $f(k) \cdot n^{O(1)}$, see [3]. The lemma implies the chain of inclusions shown in Figure 1.

¹ Details about uniformity will not be of importance in our study. We refer the interested reader to [3, 6, 9] and abbreviate “logarithmic-time-uniform” with “uniform” in the following.



■ **Figure 1** Inclusions among parallel parameterized complexity classes within para-P. An arrow from A to B means $A \subseteq B$, and a dashed arrow indicates $A \subsetneq B$. The inclusions between the two rows follow from arguments for the up-classes [3] and the other inclusions follow from the standard inclusion chain $AC^0 \subsetneq TC^0 \subseteq L \subseteq NL \subseteq AC^1 \subseteq AC^2 \subseteq P$.

▶ **Lemma 2.2.** *A problem Q is in $\text{para-AC}^{i\uparrow}$ (in the sense of Definition 2.1) iff Q can be decided by a family of Boolean circuits of depth $f(k) \cdot O(\log^i n)$ and size $f(k) \cdot n^{O(1)}$ for some computable function f .*

The advantage of our (new, more complex) definition of up-classes is that it naturally yields the classes para-L^\uparrow and para-NL^\uparrow based on para-FL and para-FNL , the parameterized versions of FL and FNL. These latter classes contain all functions $F: \Sigma^* \rightarrow \Sigma^*$ such that a Turing machine (deterministic for L, non-deterministic for NL) with a read-only input tape and a write-only output tape produces $F(w)$ on input $w \in \Sigma^*$ using only $O(\log |w|)$ cells on its work tape (in the non-deterministic case, all halting computations must lead to $F(w)$ on the output tape). It is worth noting that both FL and FNL are closed under composition (the Immerman-Szelepcsényi Theorem is needed for FNL) and that they only contain functions F with $|F(w)| \leq |w|^{O(1)}$. The parameterized function classes are defined analogously, only they contain parameterized functions $F: \Sigma^* \times \mathbb{N} \rightarrow \Sigma^* \times \mathbb{N}$ and the machines may use $f(|w|) + O(\log |w|)$ cells on the work tape on input (w, k) for some computable function $f: \mathbb{N} \rightarrow \mathbb{N}$. Note that the maximum length of $F(w)$ is now $f'(|w|) \cdot |w|^{O(1)}$ for some other computable function f' . These classes are also closed under composition.

▶ **Theorem 2.3.** *A size- k feedback vertex set can be computed in para-FL^\uparrow , if one exists.*

▶ **Lemma 2.4.** $\text{para-FNL}^{\uparrow\uparrow} = \text{para-FNL}^\uparrow$.

3 MaxSAT Variants Parameterized by Solution Size

A natural parameterization of a problem such as MAX-SAT is to take as parameter k the size of the sought solution. It is well-known that the corresponding problem $\text{p}_k\text{-MAX-SAT}$ is in para-P [34]. We prove in Section 3.1 that the problem lies in para-AC^0 and that this result generalizes to a broader range of problems. It is also known that a version with less restrictive parameter is in para-P as well [34]: $\text{p}_g\text{-MAX-SAT-ABOVE-HALF}$ asks whether there is an assignment that satisfies at least $\lceil \frac{m-m_\emptyset}{2} \rceil + g$ clauses, where m is the total number of clauses and m_\emptyset the number of empty clauses in the input. We show in Section 3.2 that this problem is strictly harder than $\text{p}_k\text{-MAX-SAT}$, as it is complete for para-TC^0 .

3.1 Maximum Bounded-Circuit Satisfiability

We consider four variants of MAX-SAT, where we maximize the number of clauses

- for $\text{p}_k\text{-MAX-SAT}$ in which at least one literal is true;
- for $\text{p}_k\text{-MAX-NAE-SAT}$ in which at least one literal is true and one is false;
- for $\text{p}_{k,d,x}\text{-MAX-EXACT-SAT}$ in which exactly x of the d literals are true;
- for $\text{p}_{k,d}\text{-MAX-DNF}$ in which all of the d literals are true.

All of these problems are special cases of Problem 3.1 below. For its definition, we say that a Boolean function $f: \{0, 1\}^n \rightarrow \{0, 1\}$ is t -robust if for every point $x \in \{0, 1\}^n$ with $f(x) = 1$ there is a set of at most t indices such that $f(y) = 1$ for any $y \in \{0, 1\}^n$ that equals x on these indices. For instance, a clause on d literals is 1-robust, while a *term* (a conjunction of literals) is d -robust. We are interested in the following promise problem:

► **Problem 3.1** ($\text{p}_{k,t}\text{-MAX-}\delta\text{-CIRCUIT-SAT}$).

Instance: Integers k and t , AC-circuits C_1, \dots, C_m , all connected to the same n input variables x_1, \dots, x_n , all with a single output gate, and all of depth at most δ .

Parameter: $k + t$

Question: Is there an assignment from the input variables to $\{0, 1\}$ such that at least k circuits evaluate to 1?

Promise: All circuits compute a t -robust function.

► **Theorem 3.2.** $\text{p}_{k,t}\text{-MAX-}\delta\text{-CIRCUIT-SAT} \in \text{para-AC}^0$.

► **Corollary 3.3.** The problems $\text{p}_k\text{-MAX-SAT}$, $\text{p}_k\text{-MAX-NAE-SAT}$, $\text{p}_{k,d,x}\text{-MAX-EXACT-SAT}$, and $\text{p}_{k,d}\text{-MAX-DNF}$ are in para-AC^0 .

3.2 Maximum Satisfiability Above Guarantee

The solution size is a very restrictive parameter for problems such as MAX-SAT, because *every* instance has *relatively large* solutions. In particular, let ϕ be a CNF with m clauses of which m_\emptyset are empty. Then ϕ *always* has an assignment that satisfies at least $\lceil \frac{m-m_\emptyset}{2} \rceil$ clauses: Pick an arbitrary assignment β and observe that either β or its bitwise complement satisfies half of the clauses [34]. Hence, $\text{p}_k\text{-MAX-SAT}$ is only interesting for large k and to obtain efficient parallel algorithms, we require a smaller parameterizations.

We start with a problem of the form $Q = \{(w, k) \mid \text{opt}(w) \leq k\}$, where $\text{opt}(w)$ is some property to be evaluated. The new problem has the form $Q' = \{(w, \pi), g \mid \pi \text{ is an easily checkable proof for } \text{opt}(w) \geq \gamma(\pi), \text{ and } \text{opt}(w) \leq \gamma(\pi) + g\}$. Here, $\gamma(\pi)$ is called the *guaranteed lower bound proved by π* or just the *guarantee*. For $Q = \text{p}_k\text{-MAX-SAT}$ the situation is particularly easy, we can take as proof π a tautology (since there is nothing to prove in this case) and set $\gamma(\pi) = \lceil \frac{m-m_\emptyset}{2} \rceil$. Note that Q' is conceptionally harder than Q : An fpt-algorithm for Q' must find a (possibly large) optimal solution, but may only use time $f(g) \cdot n^{O(1)}$ for a (possibly small) difference g .

► **Problem 3.4** ($\text{p}_g\text{-MAX-SAT-ABOVE-HALF}$).

Instance: A CNF ϕ with m clauses of which m_\emptyset are empty, and a difference $g \in \mathbb{N}$.

Parameter: g

Question: Is there an assignment that satisfies at least $\lceil \frac{m-m_\emptyset}{2} \rceil + g$ clauses?

Algorithms for above-guarantee parameterizations have led to a number of algorithmic breakthroughs, for instance in the design of algorithms for ALMOST-2SAT [35], linear-time fpt-algorithms [31], or stricter parameterizations of VERTEX-COVER [24]. One of these breakthroughs was $\text{p}_g\text{-MAX-SAT-ABOVE-HALF} \in \text{para-P}$ [34]. The following theorem sharpens this result by placing $\text{p}_g\text{-MAX-SAT-ABOVE-HALF}$ in para-TC^0 . This also pinpoints the intuition that above-guarantee parameterizations are conceptionally harder than their standard counterparts, as we obtain that $\text{p}_g\text{-MAX-SAT-ABOVE-HALF}$ is strictly harder than $\text{p}_k\text{-MAX-SAT}$ (since $\text{para-AC}^0 \subsetneq \text{para-TC}^0$).

► **Theorem 3.5.** $\text{p}_g\text{-MAX-SAT-ABOVE-HALF}$ is $\leq_{\text{tt}}^{\text{para-AC}^0}$ -complete for para-TC^0 .

This result is also tight in the sense that relaxing the parameterization further leads to an intractable problem: Let r_1, \dots, r_m be the number of literals in the clauses of a CNF ϕ , then $E(\phi) := \sum_{i=1}^m (1 - 2^{-r_i})$ is the expected number of clauses satisfied by a random truth assignment. It is well-known that an assignment that satisfies at least $E(\phi)$ clauses can be found in polynomial time [14]. However, the problem p_g -MAX-SAT-ABOVE-AVERAGE, which asks whether we can satisfy *at least* $E(\phi) + g$ clauses, is intractable:

► **Fact 3.6** ([14]). p_g -MAX-SAT-ABOVE-AVERAGE is para-NP-complete.

This result requires clauses of arbitrary size. If all clauses contain *exactly* d distinct and non-complementary literals, the problem becomes fixed-parameter tractable [2]. Note that $E(\phi) = (1 - 2^{-d})m$ holds in this case. The corresponding algorithm is quite simple and can directly be parallelized (however, it requires non-trivial results about algebraic representations of formulas that were proven in [2]; see also Section 9.2 in [16] for details).

► **Lemma 3.7.** p_g -MAX-EdSAT-ABOVE-AVERAGE \in para-AC⁰.

4 Dual Parameterizations for Variants of MaxSAT

We saw that MAX-SAT can be solved in parallel when parameterized by the solution size. However, since MAX-SAT instances always only have large solutions, we moved on to seeking solutions of size $\lceil \frac{m-m_0}{2} \rceil + g$ and then of size $E(\phi) + g$ for parameter g . We saw that the complexity increases, but also that parallel parameterized algorithms are still possible for most variants. Now, we consider *dual parameterizations* where the sought solution size is $m - k$. The corresponding problem is called p_k -ALMOST-SAT or, if the input formula comes from a family Φ , p_k -ALMOST- Φ . These problems are even harder and in order to solve them, we must, in particular, be able to decide Φ for inputs with $k = 0$:

► **Observation 4.1.** If Φ is a family of propositional formulas such that deciding satisfiability for Φ is hard for a complexity class \mathcal{C} , then p_k -ALMOST- Φ is hard for para- \mathcal{C} .

Hence we have that p_k -ALMOST-3SAT is para-NP-hard, p_k -ALMOST-HORN is para-P-hard, and p_k -ALMOST-2SAT is para-NL-hard. However, the observation does *not* provide any hint on upper bounds, e.g., it is not clear whether p_k -ALMOST-2SAT \in para-NL. Since we are interested in parallel algorithms, we study families of formulas that can be decided in subclasses of P: p_k -ALMOST-NAE-2SAT and p_k -ALMOST-2SAT in Section 4.1 (NAE-2SAT \in L and 2SAT \in NL), p_k -ALMOST-NAE-SAT(2) and p_k -ALMOST-SAT(2) in Section 4.2 (NAE-SAT(2) \in L and SAT(2) \in L), and p_k -ALMOST-DNF in Section 4.3 (DNF \in AC⁰).

4.1 Dual Parameterization for Krom Formulas

Our first result about dual parameterizations is the technically most involved part:

► **Theorem 4.2.** p_k -ALMOST-NAE-2SAT and p_k -ALMOST-2SAT both lie in para-NL[†].

The proof of the theorem is based on the well-known equivalence between p_k -ALMOST-2SAT and another member of the family of above-guarantee problems (see Section 3.2):

► **Problem 4.3** (p_g -VC-ABOVE-MATCHING).

Instance: A graph $G = (V, E)$, a matching $M \subseteq E$, a difference $g \in \mathbb{N}$.

Parameter: g

Question: Is there a set $S \subseteq V$ with $|S| \leq |M| + g$ and $e \cap S \neq \emptyset$ for every $e \in E$?

While it is known that p_k -VERTEX-COVER \in para-AC⁰ (Theorem 4.5 in [3]), we will need the rest of this section to prove the following theorem:

► **Theorem 4.4.** p_g -VC-ABOVE-MATCHING \in para-NL[↑].

Theorem 4.2 follows directly with the following lemma, which shows that the required well-known reductions [17, 35] can, firstly, be implemented in para-FAC⁰ and, secondly, the last reduction can also compute the necessary matching as part of its output.

► **Lemma 4.5.**

$$p_k\text{-ALMOST-NAE-2SAT} \leq_m^{\text{para-AC}^0} p_k\text{-ALMOST-2SAT} \leq_m^{\text{para-AC}^0} p_g\text{-VC-ABOVE-MATCHING}.$$

4.1.1 A Parallel Algorithm to Compute 0-1-Flows

Our algorithm behind Theorem 4.4 will heavily rely on repeated flow computations. Maximum flows can be computed in polynomial time with, say, the Ford–Fulkerson algorithm [22]. However, computing the value of a weighted maximum flow is P-complete [27], and whether we can compute a 0-1-flow in parallel is a long standing open problem [33]. It is worth noting that a maximum 0-1-flow can be computed in randomized NC via a reduction to the maximum matching problem in bipartite graphs [33]. Unfortunately, this reduction is *not* parameter-preserving and, thus, we may not apply parameterized matching algorithms [5].

Our objective in this section is to show that a flow of value k can be computed in parallel; more precisely, that there is a function in para-FNL[↑] mapping $((G, s, t), k)$ to a 0-1-flow of value k from s to t , if it exists, and otherwise to a maximum flow (formally, the output of a parameterized function must be a pair where the second component is a new parameter value, but we will not need this here and just silently assume that this value is set to, say, 0).

Computing Paths in FNL. It is well-known that the reachability problem in digraphs is the canonical complete problem for NL and, thus, it may seem trivial that we should be able to compute paths in FNL. However, being able to tell whether there is a path from s to t is not the same as actually finding such a path: For instance, it is known that in tournaments (digraphs with exactly one edge between any pair of vertices) reachability lies in AC⁰, the distance problem is NL-complete, and constructing a path longer than the shortest path by a factor of $1 + \epsilon$ can be done in deterministic logarithmic space [37] – meaning that reachability and path construction can have vastly different complexities. Nevertheless:

► **Lemma 4.6.** *There is a function in FNL that maps (G, s, t) to a shortest path from s to t , provided it exists.*

Computing 0-1-Flows in para-FNL[↑]. The most important operation in the Ford–Fulkerson algorithm is the computation of an *augmenting path*. An iterated application of Lemma 4.6 therefore allows us to compute a small flow:

► **Theorem 4.7.** *There is a parameterized function in para-FNL[↑] that maps $((G, s, t), k)$ to a flow from s to t in G of value k , if it exists, or to a maximum flow otherwise.*

Let p_k -FLOW = $\{((G, s, t), k) \mid \text{there is a 0-1-flow } f \text{ from } s \text{ to } t \text{ in } G \text{ with } |f| \leq k\}$ be the corresponding parameterized decision problem.

► **Corollary 4.8.** p_k -FLOW \in para-NL[↑].

19:10 On the Parallel Parameterized Complexity of MaxSAT Variants

The following corollary observes that instead of starting with the empty flow we can also start with an arbitrary flow f and augment it k times:

► **Corollary 4.9.** *There is a parameterized function in para-FNL^\uparrow that maps $((G, s, t, f), k)$, where f is an s - t -flow in G , to an s - t -flow f' in G of value $|f| + k$, if it exists, or to a maximum flow otherwise.*

► **Remark 4.10.** While Theorem 4.7 and the corollaries 4.8 and 4.9 only speak about 0-1-flows, it is easy to see that the same techniques can be used to compute flows in networks with *fixed constant capacities*: just replace each edge with capacity c by c parallel edges and divide each of these edges with a fresh vertex afterwards. In particular, Corollary 4.9 can also be used to augment *half-integral flows* in networks with fixed maximum capacity.

4.1.2 Linear Programs for Vertex Cover and Matching

To prove Theorem 4.4, we will study a more general problem and obtain the theorem as a simple corollary: Instead of using matchings as proofs for lower bounds for the vertex cover problem, we use fractional solutions of LP-relaxations. Let us fix some notations: For a linear program Π let $\text{vars}(\Pi)$ be the set of variables occurring in Π . A *solution* for Π is an assignment $\alpha: \text{vars}(\Pi) \rightarrow \mathbb{Q}$ that satisfies all inequalities, and the *solution value* (or just *value*) $|\alpha|$ of α is the value of the optimization function under α . An *optimal solution* is an assignment that has the minimum (or maximum) solution value over all possible assignments. We say an assignment α is *integral* if $\alpha: \text{vars}(\Pi) \rightarrow \mathbb{N}$ for all $x \in \text{vars}(\Pi)$; α is *half-integral* if all $\alpha(x)$ are half-integral, meaning $\alpha(x) = i/2$ for some $i \in \mathbb{N}$; otherwise α is *fractional*. Let $\text{opt}_{\mathbb{Q}}(\Pi)$, $\text{opt}_{\mathbb{N}/2}(\Pi)$, and $\text{opt}_{\mathbb{N}}(\Pi)$ denote the optimal value of a fractional, half-integral, and integral solution for Π , respectively. We are interested in the following two linear programs:

► **Definition 4.11** (Linear Program $\Pi_{\text{VC}}(G)$ for Vertex Cover of a Graph $G = (V, E)$).

$$\begin{aligned} \text{Minimize } \sum_{v \in V} x_v \text{ subject to } & x_u + x_v \geq 1 \quad \text{for all } \{u, v\} \in E, \\ & 0 \leq x_v \leq 1 \quad \text{for all } v \in V. \end{aligned}$$

► **Definition 4.12** (Linear Program $\Pi_{\text{M}}(G)$ for Matching of a Graph $G = (V, E)$).

$$\begin{aligned} \text{Maximize } \sum_{e \in E} y_e \text{ subject to } & \sum_{v \in e} y_e \leq 1 \quad \text{for all } v \in V, \\ & 0 \leq y_e \leq 1 \quad \text{for all } e \in E. \end{aligned}$$

A vertex cover of G naturally corresponds to an integral solution $\alpha_{\mathbb{N}}$ of $\Pi_{\text{VC}}(G)$ and a matching corresponds to an integral solution $\beta_{\mathbb{N}}$ of $\Pi_{\text{M}}(G)$ (the index “ \mathbb{N} ” emphasizes that the solution is integral). In particular, $\text{opt}_{\mathbb{N}}(\Pi_{\text{VC}}(G))$ and $\text{opt}_{\mathbb{N}}(\Pi_{\text{M}}(G))$ are the sizes of a minimum vertex cover and a maximum matching of G , respectively. The programs are dual to each other, which implies that their optimal fractional solutions have the same value.

► **Fact 4.13** (Nemhauser-Trotter Theorem [36], [16, Chapter 2]). *Let $G = (V, E)$ be a graph. Then $\Pi_{\text{VC}}(G)$ and $\Pi_{\text{M}}(G)$ have solutions α and β , respectively, with the following properties:*

1. $\text{opt}_{\mathbb{Q}}(\Pi_{\text{VC}}(G)) = |\alpha| = |\beta| = \text{opt}_{\mathbb{Q}}(\Pi_{\text{M}}(G))$,
2. α and β are half-integral,
3. there is an optimal integral solution γ for $\Pi_{\text{VC}}(G)$ such that for $v \in V$ with $\alpha(x_v) \neq 1/2$ we have $\gamma(x_v) = \alpha(x_v)$ (that is, γ equals α on its integral part).

Fact 4.13 implies that the following (in)equalities hold, where $\alpha_{\mathbb{N}}$ and $\alpha_{\mathbb{N}/2}$ are arbitrary integral and half-integral solutions for $\Pi_{\text{VC}}(G)$ and $\beta_{\mathbb{N}}$ and $\beta_{\mathbb{N}/2}$ correspondingly for $\Pi_{\text{M}}(G)$:

$$\begin{aligned} |\beta_{\mathbb{N}}| \leq \text{opt}_{\mathbb{N}}(\Pi_{\text{M}}(G)) & \stackrel{(*)}{\leq} \overbrace{\text{opt}_{\mathbb{N}/2}(\Pi_{\text{M}}(G))}^{(*)} = \text{opt}_{\mathbb{N}/2}(\Pi_{\text{VC}}(G)) \stackrel{(**)}{\leq} \overbrace{\text{opt}_{\mathbb{N}}(\Pi_{\text{VC}}(G))}^{(**)} \leq |\alpha_{\mathbb{N}}| \\ & \stackrel{(**)}{\leq} |\beta_{\mathbb{N}/2}| \stackrel{(**)}{\leq} |\alpha_{\mathbb{N}/2}| \end{aligned} \quad (1)$$

The parameter of p_g -VC-ABOVE-MATCHING is the difference between the upper left value $|\beta_{\mathbb{N}}|$, which is the size of some matching of G , and (**), which is the size of a minimum vertex cover of G . When working with linear programs, it is natural to work with a different (“better”) parameter, namely the difference between the lower left value $|\beta_{\mathbb{N}/2}|$ and (**):

► **Problem 4.14** (p_g -VC-ABOVE-RELAXED-MATCHING).

Instance: A graph $G = (V, E)$, a half-integral solution $\beta_{\mathbb{N}/2}$ for $\Pi_M(G)$, and a number g .

Parameter: g

Question: Is there a set $S \subseteq V$ with $|S| \leq |\beta_{\mathbb{N}/2}| + g$ and $e \cap S \neq \emptyset$ for every $e \in E$?

4.1.3 An FPT-Algorithm for Solving VC Above Half-Integral Matching

Let us briefly review how one usually shows p_g -VC-ABOVE-RELAXED-MATCHING \in para-P:

Step 0: Computing an Optimal Half-Integral Solution. Compute an optimal half-integral solution α for $\Pi_{VC}(G)$ in polynomial-time ($|\alpha|$ has the value (*) in (1)).

Step 1: Reduction to the All-1/2-Solution. We turn α into an “all-1/2-solution,” meaning that $\alpha(x_v) = 1/2$ holds for all vertices. To achieve this, we use Fact 4.13, which tells us that vertices $v \in V$ with $\alpha(x_v) = 0$ are not part of an optimal vertex cover while vertices with $\alpha(x_v) = 1$ are. Thus, we can delete all these vertices and continue with the same parameter g (the integrality excess does not change). Note that α restricted to the new graph (which we still call G) is constantly $1/2$, which we denote as $\alpha \equiv 1/2$.

Step 2: Making the All-1/2-Solution Unique. Now $\alpha \equiv 1/2$ is an optimal solution, but there may be other optimal half-integral solutions. (For instance, the all-1/2-solution is an optimal solution for any even cycle, but so is the integral solution $\alpha(i) = (i \bmod 2)$.) We can check in polynomial time whether α is the unique optimal solution as follows: Test for every x_v whether $\text{opt}_{\mathbb{Q}}(\Pi_{VC}(G)) = \text{opt}_{\mathbb{Q}}(\Pi_{VC}(G - \{v\})) + 1$. If so, there is an optimal solution other than α that assigns 1 to x_v . We remove v from G using Fact 4.13, leave g untouched, and repeat until the all-1/2-solution is the only optimal solution.

Step 3: Branching. Suppose we knew that some vertex $v \in V$ is part of an optimal vertex cover of G . Then $\text{opt}_{\mathbb{N}}(G - \{v\}) = \text{opt}_{\mathbb{N}}(G) - 1$ while $\text{opt}_{\mathbb{Q}}(G - \{v\}) = \text{opt}_{\mathbb{Q}}(G) - 1/2$. This means that the integrality excess of $G - \{v\}$ is reduced by $1/2$ compared to G . Of course, we do not know which vertices are part of an optimal vertex cover, but we can find them using *branching*: Pick an arbitrary edge $\{u, v\} \in E$ and recursively run the whole algorithm (starting from Step 1 once more) for $G - \{u\}$ and $G - \{v\}$, but now for the parameter $g - 1/2$ (the parameter should actually be an integer, but it is convenient for the recursion to allow integers divided by 2 as parameters in this setting).

It is now easy to see that the depth of the search tree of the above algorithm is $2g$, so the total runtime is $4^g \cdot n^{O(1)}$.

4.1.4 A Parallel Algorithm for VC Above Half-Integral Matching

In this section we parallelize the different steps sketched above for solving Problem 4.14. This yields the following theorem, of which Theorem 4.4 is a corollary:

► **Theorem 4.15.** p_g -VC-ABOVE-RELAXED-MATCHING \in para-NL $^{\uparrow}$.

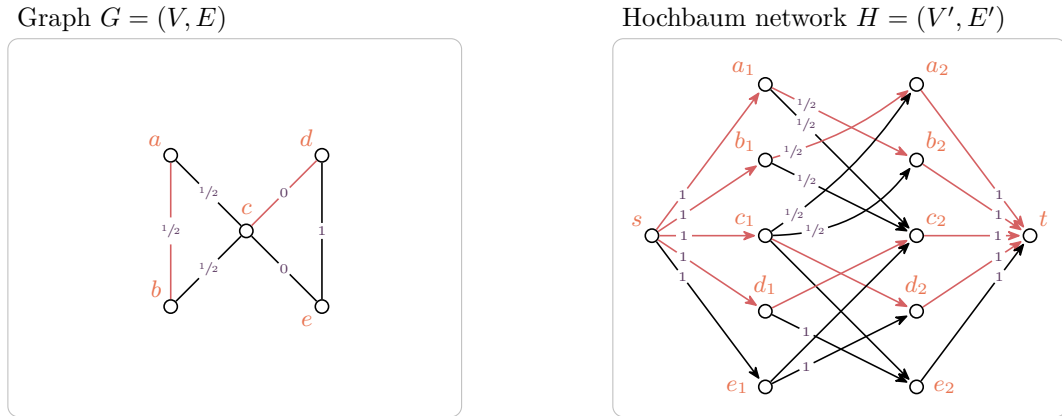
While steps 1 and 3 are easy to parallelize (search trees can be traversed in parallel), steps 0 and 2 are not. They either involve open problems (like computing optimal solutions for $\Pi_{VC}(G)$ in parallel) or are very sequential (like the iterative removal of vertices in step 2).

Parallelizing Step 0: Computing an Optimal Half-Integral Solution. Given a half-integral solution β of $\Pi_M(G)$, we wish to compute an optimal half-integral solution α of $\Pi_{VC}(G)$. A para-P-machine could just ignore β and solve the linear program, but we only have a para-NL[†]-machine. The core idea we use was developed by Iwata, Oka, and Yoshida [31] in the context of a linear-time algorithm: One can encode an (optimal) solution of $\Pi_M(G)$ into a (maximum) flow in the so-called *Hochbaum network*. More crucially, we can obtain an (optimal) solution for $\Pi_M(G)$ and $\Pi_{VC}(G)$ from a (maximum) flow in this network.

In detail, for a graph $G = (V, E)$ the *Hochbaum network* is the digraph $H = (V', E')$ with V' consisting of $V_1 = \{v_1 \mid v \in V\}$ and $V_2 = \{v_2 \mid v \in V\}$ plus the two vertices s and t . The edge set is $E' = \{(s, v_1) \mid v \in V\} \cup \{(u_1, v_2) \mid \{u, v\} \in E\} \cup \{(v_2, t) \mid v \in V\}$, i. e., from s we get to all vertices in V_1 , then we can cross from u_1 to v_2 exactly if $\{u, v\} \in E$ (and then also from v_1 to u_2), and from all vertices in V_2 we can get to t .

- **Fact 4.16** ([30, 31]). *Let $G = (V, E)$ be a graph and $H = (V', E')$ be its Hochbaum network.*
1. *If β is a solution of $\Pi_M(G)$, then the mapping $f_\beta(s, v_1) = f_\beta(v_2, t) = \sum_{w \in N(v)} \beta(y_{\{v,w\}})$ and $f_\beta(u_1, v_2) = f_\beta(v_1, u_2) = \beta(y_{\{u,v\}})$ is an s - t -flow with $|f_\beta| = 2|\beta|$ in H .*
 2. *If f is an s - t -flow in H , then $\beta_f(y_{\{u,v\}}) = \frac{1}{2}(f(u_1, v_2) + f(v_1, u_2))$ is a solution for $\Pi_M(G)$ with $|\beta_f| = |f|/2$.*

Note that, in particular, β is an optimal solution of $\Pi_M(G)$ iff f_β is maximal and, *vice versa*, f is a maximal flow iff β_f is an optimal solution. Figure 2 illustrates these definitions and the interplay between solutions for $\Pi_M(G)$ and flows in the corresponding Hochbaum network. Since the translation between flows and solutions is computationally easy, we freely switch between flows and solutions for $\Pi_M(G)$ as needed.



■ **Figure 2** The left side shows a graph $G = (V, E)$ on five vertices $V = \{a, b, c, d, e\}$. On the edges a *half-integral* solution β of $\Pi_M(G)$ of value $\text{opt}_{N/2}(\Pi_M(G)) = 2.5$ is illustrated. The two red edges constitute an optimal *integral* solution for $\Pi_M(G)$. The right side shows the Hochbaum network $H = (V', E')$ corresponding to G . The edges are labeled with a maximum flow f_β of value $|f_\beta| = 5$ that corresponds to β . The integral solution (the red maximum matching) corresponds to the flow of value four that sends one unit over every red edge (which is not maximal).

► **Lemma 4.17.** *There is a function in para-FNL[†] that maps $((G, \beta), g)$, consisting of a graph G , a half-integral solution β of $\Pi_M(G)$, and a number g , to an optimal half-integral solution β' of G , provided such a solution with $|\beta'| \leq |\beta| + g$ exists.*

Lemma 4.17 provides a reduction rule for p_g -VC-ABOVE-RELAXED-MATCHING: We can map $((G, \beta), g)$ to $((G, \beta'), g - |\beta'| + |\beta|)$ such that β' is optimal. Since we will often use triples (G, H, f) where $G = (V, E)$ is a graph, $H = (V', E')$ is its Hochbaum network, and f is a maximum flow in H , let us call such a triple a *graph-Hochbaum-flow triple*.

We now have a way of computing an optimal solution β' for $\Pi_M(G)$, but for the next steps of our algorithm, we need a half-integral solution α of $\Pi_{VC}(G)$. Fortunately, there is another observation that shows how a maximum flow f can be used to derive an optimal solution α_f for $\Pi_{VC}(G)$ (note that this solution is trivially half-integral):

► **Fact 4.18** ([30, 31]). *Let (G, H, f) be a graph-Hochbaum-flow triple. Let $X \subseteq V'$ be the set of vertices reachable from s in the residual network R_f . Then*

$$\alpha_f(x_v) = \begin{cases} 0 & \text{if } v_1 \in X \text{ and } v_2 \notin X, \\ 1 & \text{if } v_1 \notin X \text{ and } v_2 \in X, \\ 1/2 & \text{otherwise,} \end{cases}$$

is an optimal solution for $\Pi_{VC}(G)$.

► **Lemma 4.19.** *There is a function in FNL that maps (G, β) , consisting of a graph and an optimal half-integral solution of $\Pi_M(G)$, to an optimal half-integral solution α of $\Pi_{VC}(G)$.*

Together, Lemmas 4.17 and 4.19 clearly allow us to perform Step 0 of the computation (namely the computation of an optimal solution α of $\Pi_{VC}(G)$) using a para-FNL[†]-machine.

Parallelizing Step 1: Reduction to the All-1/2-Solution. The next step turns the half-integral solution α into an all-1/2-solution by deleting all vertices v for which $\alpha(x_v) \neq 1/2$. Clearly, this can be done in parallel. Note that here we really need an optimal solution α of $\Pi_{VC}(G)$ rather than a solution β of $\Pi_M(G)$: Only α tells us which vertices can be removed.

Parallelizing Step 2: Making the All-1/2-Solution Unique. The sequential method described in Section 4.1.3 for implementing Step 2 is exactly that: highly *sequential*. It is not difficult to construct a graph for which the number of iterations used by this method is linear in the graph size – just consider a large matching: The all-1/2-solution is an optimal solution, but in each iteration of Step 2 only one edge will be removed from the graph. Even worse, after the removal of a vertex it might be necessary to recompute the optimal solution α .

For a parallel algorithm, we need some further insights from the work of Iwata, Oka, and Yoshida [31]. Let us start with some definitions, which adapt their ideas to our context:

► **Definition 4.20.** *Let (G, H, f) be a graph-Hochbaum-flow triple. A set $S \subseteq V'$ is loose if the following holds:*

1. S is a strongly connected component of the residual graph $R_f = (V', E'_f)$.
2. $\{v \in V' \mid v_1 \in S\}$ and $\{v \in V' \mid v_2 \in S\}$ are disjoint.

We call a loose set removable, if the following holds additionally:

3. *There are no edges leaving S in R_f , i. e., no edges $(x, y) \in E'_f$ with $x \in S$ and $y \notin S$.*

► **Definition 4.21.** *Let (G, H, f) be a graph-Hochbaum-flow triple and $S \subseteq V'$ be a removable set. Removing S yields the following triple (G^{-S}, H^{-S}, f^{-S}) :*

1. $G^{-S} = G - \{v \in V' \mid v_1 \in S \vee \exists w \in N(v)(w_1 \in S)\}$,
2. $H^{-S} = H - S$,
3. f^{-S} is the flow induced on the vertices of H^{-S} .

Intuitively, (G^{-S}, H^{-S}, f^{-S}) should also be a graph-Hochbaum-flow triple and this is the case, at least if $\alpha \equiv 1/2$ is an optimal solution:

► **Fact 4.22** ([31, Corollary 4.2 and the subsequent discussion]). *Let (G, H, f) be a graph-Hochbaum-flow triple such that $\alpha \equiv 1/2$ is an optimal solution of $\Pi_{\text{VC}}(G)$.*

1. *If there is no removable set S , then $\alpha \equiv 1/2$ is the only optimal solution for $\Pi_{\text{VC}}(G)$.*
2. *If there is a removable set S , then (G^{-S}, H^{-S}, f^{-S}) is a graph-Hochbaum-flow triple and G^{-S} has the same integrality excess as G .*

While the fact tells us which vertices we should remove from G , it does not tell us which will be part of the vertex cover. This can easily be fixed, however: When S is removed, we can set $\beta(x_v) = 0$ for all $v_1 \in S$ and $\beta(x_v) = 1$ for all $v \in V$ for which there is a $w \in N(v)$ with $w_1 \in S$, see the discussion after Lemma 4.6 in [31] for details.

Using Fact 4.22, an NL-machine can test whether $\alpha \equiv 1/2$ is the only optimal solution of $\Pi_{\text{VC}}(G)$ by looking for a removable S . Furthermore, the machine can iteratively remove such sets until the all-1/2-solution is the only optimal half-integral solution. This may seem similarly sequential as the repetitive removal of vertices in Step 2, but it turns out that we can remove everything in a single run:

► **Lemma 4.23.** *There is a function in FNL that gets a graph-Hochbaum-flow triple (G, H, f) as input and outputs the graph-Hochbaum-flow triple (G^-, H^-, f^-) resulting from iteratively removing removable sets as long as they exist.*

Parallelizing Step 3: Branching. As mentioned earlier, the branching step is easy to parallelize, as the two children in the search tree can be explored in parallel. Branching also fits nicely into our framework of the up-class para-FNL^\uparrow , which arises from parameter-dependent-many iterations of a linear function in para-FNL : In each iteration a list of instances is on the input tape and this list is mapped to at most twice as many new instances on the output tape, but with a reduction of the parameter in all these instances.

This completes our description of Theorem 4.4. A formal proof that glues together all the ingredients developed within this section can be found in the technical report.

4.2 Dual Parameterization When Every Variables Occur at Most Twice

A formula ϕ is in $\text{CNF}(2)$ if it is a CNF and every variable occurs at most twice (variables may occur positively and negatively, and clauses may be arbitrary large). Johannsen showed that the satisfiability problem and the nae-satisfiability problem for $\text{CNF}(2)$ formulas are complete for L [32]. We extend this result and observe that the logspace algorithms can be modified such that they solve the corresponding maximization problem: Given a $\text{CNF}(2)$ formula ϕ , they output the maximum number of simultaneously satisfiable clauses. Combined with Observation 4.1 we obtain:

► **Theorem 4.24.** $\text{p}_k\text{-ALMOST-NAE-SAT}(2)$ and $\text{p}_k\text{-ALMOST-SAT}(2)$ are complete for para-L .

► **Lemma 4.25.** *There is a function in FL that maps $\text{CNF}(2)$ formulas ϕ to the maximum number of simultaneously satisfiable clauses of ϕ .*

► **Lemma 4.26.** *There is a function in FL that maps $\text{CNF}(2)$ formulas ϕ to the maximum number of simultaneously nae-satisfiable clauses of ϕ .*

4.3 Dual Parameterization for Formulas in Disjunctive Normal Form

Testing whether a DNF is satisfiable can be done in polynomial time (even in AC^0), in contrast, deciding whether we can satisfy k terms simultaneously (i. e., MAX-DNF) is NP-complete [20]. In this section we study MAX-DNF with a dual parameterization: p_k -ALMOST-DNF asks whether a given DNF ϕ has an assignment that satisfies at least $m - k$ terms.

► **Theorem 4.27.** p_k -ALMOST-DNF \in para- AC^0 .

The proof of the theorem boils down to the following reduction and the subsequent lemma. Construct a CNF ψ from ϕ by simply negating every term, i. e., if $(\ell_1 \wedge \dots \wedge \ell_d)$ is a term in ϕ , we add $(\neg \ell_1 \vee \dots \vee \neg \ell_d)$ as clause to ψ . Observe that every assignment that satisfies a term in ϕ does *not* satisfy the correspond clause in ψ . Hence, there is an assignment satisfying at least $m - k$ terms in ϕ if there is an assignment that satisfies *at most* k clauses in ψ . In other words, we have reduced p_k -ALMOST-DNF to p_k -MIN-SAT.

► **Lemma 4.28.** p_k -MIN-SAT \in para- AC^0 .

5 Structural Parameterizations for Partial MaxSAT Variants

The most general incarnation of MAX-SAT is the *partially weighted version*: We are given a CNF ϕ and a weight function $\omega: \text{clauses}(\phi) \rightarrow \mathbb{N} \cup \{\infty\}$, in which we call clauses C *soft* if $\omega(C) < \infty$ and *hard* otherwise. The goal is to find among all assignments $\beta: \text{vars}(\phi) \rightarrow \{0, 1\}$ that satisfy *all* hard clauses the one that *maximises* the sum of the satisfied soft clauses. We refer to the decision version, in which a target sum is given, as PARTIAL-MAX-SAT.

The usual approach to identify tractable fragments of PARTIAL-MAX-SAT is to use *structural parameters*, see [18] for an overview. Structural parameters are defined over the *incidence graph* of the input formula ϕ , which is the bipartite graph on vertex set $\text{vars}(\phi) \cup \text{clauses}(\phi)$ that contains an edge between $x \in \text{vars}(\phi)$ and $C \in \text{clauses}(\phi)$ if either $x \in C$ or $\neg x \in C$.

Natural parameters are the *vertex cover number*, the *treedepth*, the *feedback vertex set number*, or the *treewidth* of the incidence graph. See Figure 3 for an overview of how these parameters are related. It is well-known that PARTIAL-MAX-SAT is in para-P parameterized by any of these, which follows quite directly from optimization versions of Courcelle's Theorem [12]. By the parallel version of this theorem [4] it follows that PARTIAL-MAX-SAT lies in para- $AC^{2\uparrow}$ if parameterized by *both*, the structural parameter and the solution size.

In the remainder of this section we develop handcrafted algorithms for all four structural parameters that (i) work independently of the solution size (it does *not* have to be a parameter), (ii) work with arbitrary weights, and (iii) are constructive in the sense that an optimal assignment is output. Figure 3 reveals intriguing connections between these parameters to the degree of parallelism we can achieve – a detail that is usually concealed in the study of sequential para-P algorithms.

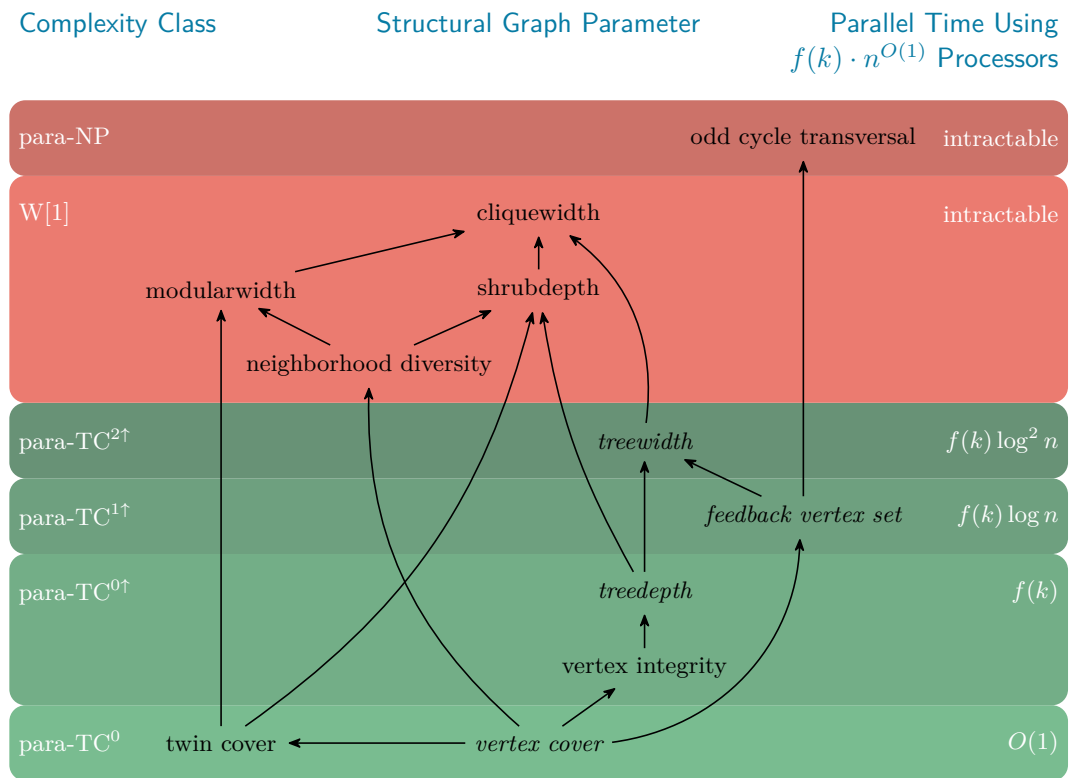
► **Theorem 5.1.** p_{vc} -PARTIAL-MAX-SAT \in para- TC^0 , p_{td} -PARTIAL-MAX-SAT \in para- $TC^{0\uparrow}$, p_{fvs} -PARTIAL-MAX-SAT \in para- $TC^{1\uparrow}$, p_{tw} -PARTIAL-MAX-SAT \in para- $AC^{2\uparrow}$.

6 Conclusion and Outlook

We presented a comprehensive list of parallel fixed-parameter algorithms for variations of MAX-SAT. As highlight we presented the first parallel algorithms for p_k -ALMOST-NAE-2SAT and p_k -ALMOST-2SAT, which implies parallel fpt-algorithms for various problems such as the odd cycle transversal problem.

The central method for proving that the latter problem is fixed-parameter tractable – the iterative compression method – seems to be inherently sequential. Interestingly, our parallel algorithm builds on another method that seems inherently sequential in general, namely the computation of maximum flows. However, using properties of the Hochbaum network allowed us to break the computation of a maximum flow into a series of small flow computations, which we then can perform in parallel using fpt -many parallel processing units.

We remark that from a complexity-theoretic point of view, p_k -ALMOST-2SAT is a harder problem than p_k -ALMOST-NAE-2SAT as the former is easily seen to be hard for para-NL while the latter is easily seen to lie in para-WL (see [19] for a discussion of these classes), which suggests that the problems have different complexity. As open problem we thus leave the question of whether p_k -ALMOST-NAE-2SAT \in para- L^\uparrow holds (which would imply that the odd cycle transversal problem lies in this class, too). While we know of no complexity-theoretic assumption that would contradict this, our proofs make heavy use of finding augmenting paths in networks and these networks seem to be inherently directed.



■ **Figure 3** A Hasse diagram of the major structural graph parameters. An arrow from A to B here means that for any graph G the parameter B is upper-bounded by a function in A . Each node corresponds to the complexity of PARTIAL-MAX-SAT parameterized by the corresponding value of the input’s incidence graph. The emphasized entries are the results proven in Theorem 5.1. The remaining parameters are briefly discussed in the technical report.

References

- 1 Faisal N. Abu-Khzam and Karam Al Kontar. A Brief Survey of Fixed-Parameter Parallelism. *Algorithms*, 13(8):197, 2020. doi:10.3390/a13080197.
- 2 Noga Alon, Gregory Z. Gutin, Eun Jung Kim, Stefan Szeider, and Anders Yeo. Solving MAX- r -SAT Above a Tight Lower Bound. *Algorithmica*, 61(3):638–655, 2011. doi:10.1007/s00453-010-9428-7.

- 3 Max Bannach, Christoph Stockhusen, and Till Tantau. Fast Parallel Fixed-Parameter Algorithms via Color Coding. In *10th International Symposium on Parameterized and Exact Computation, IPEC 2015, September 16-18, 2015, Patras, Greece*, volume 43 of *LIPICs*, pages 224–235. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2015. doi:10.4230/LIPICs.IPEC.2015.224.
- 4 Max Bannach and Till Tantau. Parallel Multivariate Meta-Theorems. In *11th International Symposium on Parameterized and Exact Computation, IPEC 2016, August 24-26, 2016, Aarhus, Denmark*, volume 63 of *LIPICs*, pages 4:1–4:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPICs.IPEC.2016.4.
- 5 Max Bannach and Till Tantau. Computing Kernels in Parallel: Lower and Upper Bounds. In *13th International Symposium on Parameterized and Exact Computation, IPEC 2018, August 20-24, 2018, Helsinki, Finland*, volume 115 of *LIPICs*, pages 13:1–13:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPICs.IPEC.2018.13.
- 6 David A. Mix Barrington, Neil Immerman, and Howard Straubing. On Uniformity within NC^1 . *Journal of Computer and System Sciences*, 41(3):274–306, 1990. doi:10.1016/0022-0000(90)90022-D.
- 7 Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors. *Handbook of Satisfiability, Second Edition*. IOS Press, 2021.
- 8 Liming Cai, Jianer Chen, Rodney G. Downey, and Michael R. Fellows. Advice Classes of Parameterized Tractability. *Annals of Pure and Applied Logic*, 84(1):119–138, 1997. doi:10.1016/S0168-0072(95)00020-8.
- 9 Yijia Chen and Jörg Flum. Some Lower Bounds in Parameterized AC^0 . In *41st International Symposium on Mathematical Foundations of Computer Science, MFCS 2016, August 22-26, 2016 - Kraków, Poland*, pages 27:1–27:14, 2016. doi:10.4230/LIPICs.MFCS.2016.27.
- 10 Yijia Chen and Jörg Flum. Parameterized Parallel Computing and First-Order Logic. In *Fields of Logic and Computation III - Essays Dedicated to Yuri Gurevich on the Occasion of His 80th Birthday*, pages 57–78, 2020. doi:10.1007/978-3-030-48006-6_5.
- 11 Yijia Chen, Jörg Flum, and Xuanguai Huang. Slicewise Definability in First-Order Logic with Bounded Quantifier Rank. In *26th EACSL Annual Conference on Computer Science Logic, CSL 2017, August 20-24, 2017, Stockholm, Sweden*, volume 82 of *LIPICs*, pages 19:1–19:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPICs.CSL.2017.19.
- 12 Bruno Courcelle. The Monadic Second-Order Logic of Graphs. I. Recognizable Sets of Finite Graphs. *Inf. Comput.*, 85(1):12–75, 1990. doi:10.1016/0890-5401(90)90043-H.
- 13 Robert Crowston, Michael R. Fellows, Gregory Z. Gutin, Mark Jones, Frances A. Rosamond, Stéphan Thomassé, and Anders Yeo. Simultaneously Satisfying Linear Equations Over \mathbb{F}_2 : MaxLin2 and Max-r-Lin2 Parameterized Above Average. In *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2011, December 12-14, 2011, Mumbai, India*, pages 229–240, 2011. doi:10.4230/LIPICs.FSTTCS.2011.229.
- 14 Robert Crowston, Gregory Z. Gutin, Mark Jones, Venkatesh Raman, and Saket Saurabh. Parameterized Complexity of MaxSat Above Average. *Theor. Comput. Sci.*, 511:77–84, 2013. doi:10.1016/j.tcs.2013.01.005.
- 15 Robert Crowston, Gregory Z. Gutin, Mark Jones, and Anders Yeo. A New Lower Bound on the Maximum Number of Satisfied Clauses in MaxSAT and Its Algorithmic Applications. *Algorithmica*, 64(1):56–68, 2012. doi:10.1007/s00453-011-9550-1.
- 16 Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. doi:10.1007/978-3-319-21275-3.
- 17 Marek Cygan, Marcin Pilipczuk, Michal Pilipczuk, and Jakub Onufry Wojtaszczyk. On Multiway Cut Parameterized Above Lower Bounds. *ACM Transactions on Computation Theory*, 5(1):3:1–3:11, 2013. doi:10.1145/2462896.2462899.

- 18 Holger Dell, Eun Jung Kim, Michael Lampis, Valia Mitsou, and Tobias Mömke. Complexity and Approximability of Parameterized MAX-CSPs. *Algorithmica*, 79(1):230–250, 2017. doi:10.1007/s00453-017-0310-8.
- 19 Michael Elberfeld, Christoph Stockhusen, and Till Tantau. On the Space and Circuit Complexity of Parameterized Problems: Classes and Completeness. *Algorithmica*, 71(3):661–701, 2015. doi:10.1007/s00453-014-9944-y.
- 20 Bruno Escoffier and Vangelis Th. Paschos. Differential Approximation of MinSAT, MaxSAT and Related Problems. In *Computational Science and Its Applications - ICCSA 2005, International Conference, Singapore, May 9-12, 2005, Proceedings, Part IV*, pages 192–201, 2005. doi:10.1007/11424925_22.
- 21 Jörg Flum and Martin Grohe. Describing Parameterized Complexity Classes. *Information and Computation*, 187(2):291–319, 2003. doi:10.1016/S0890-5401(03)00161-5.
- 22 L. R. Ford and D. R. Fulkerson. Maximal Flow Through a Network. *Canadian Journal of Mathematics*, 8:399–404, 1956. doi:10.4153/CJM-1956-045-5.
- 23 M. R. Garey, David S. Johnson, and Larry J. Stockmeyer. Some Simplified NP-Complete Graph Problems. *Theor. Comput. Sci.*, 1(3):237–267, 1976. doi:10.1016/0304-3975(76)90059-1.
- 24 Shivam Garg and Geevarghese Philip. Raising The Bar For Vertex Cover: Fixed-Parameter Tractability Above a Higher Guarantee. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 1152–1166. SIAM, 2016. doi:10.1137/1.9781611974331.ch80.
- 25 Serge Gaspers and Stefan Szeider. Kernels for Global Constraints. In *IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, July 16-22, 2011*, pages 540–545, 2011. doi:10.5591/978-1-57735-516-8/IJCAI11-098.
- 26 Serge Gaspers and Stefan Szeider. Guarantees and Limits of Preprocessing in Constraint Satisfaction and Reasoning. *Artif. Intell.*, 216:1–19, 2014. doi:10.1016/j.artint.2014.06.006.
- 27 Leslie M. Goldschlager, Ralph A. Shaw, and John Staples. The Maximum Flow Problem is Log Space Complete for P. *Theoretical Computer Science*, 21:105–111, 1982. doi:10.1016/0304-3975(82)90092-5.
- 28 Martin Grohe. The Structure of Tractable Constraint Satisfaction Problems. In *Mathematical Foundations of Computer Science 2006, 31st International Symposium, MFCS 2006, Stará Lesná, Slovakia, August 28-September 1, 2006, Proceedings*, pages 58–72, 2006. doi:10.1007/11821069_5.
- 29 Gregory Z. Gutin, Mark Jones, Dominik Scheder, and Anders Yeo. A new Bound for 3-Satisfiable MaxSat and its Algorithmic Application. *Inf. Comput.*, 231:117–124, 2013. doi:10.1016/j.ic.2013.08.008.
- 30 Dorit S. Hochbaum. Solving Integer Programs over Monotone Inequalities in three Variables: A Framework for Half Integrality and Good Approximations. *European Journal of Operational Research*, 140(2):291–321, 2002. doi:10.1016/S0377-2217(02)00071-1.
- 31 Yoichi Iwata, Keigo Oka, and Yuichi Yoshida. Linear-Time FPT Algorithms via Network Flow. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 1749–1761. SIAM, 2014. doi:10.1137/1.9781611973402.127.
- 32 Jan Johannsen. Satisfiability Problems Complete for Deterministic Logarithmic Space. In *STACS 2004, 21st Annual Symposium on Theoretical Aspects of Computer Science, Montpellier, France, March 25-27, 2004, Proceedings*, pages 317–325, 2004. doi:10.1007/978-3-540-24749-4_28.
- 33 Richard M. Karp, Eli Upfal, and Avi Wigderson. Constructing a Perfect Matching is in Random NC. *Combinatorica*, 6(1):35–48, 1986. doi:10.1007/BF02579407.
- 34 Meena Mahajan and Venkatesh Raman. Parameterizing above Guaranteed Values: MaxSat and MaxCut. *J. Algorithms*, 31(2):335–354, 1999. doi:10.1006/jagm.1998.0996.

- 35 N. S. Narayanaswamy, Venkatesh Raman, M. S. Ramanujan, and Saket Saurabh. LP can be a cure for Parameterized Problems. In *29th International Symposium on Theoretical Aspects of Computer Science, STACS 2012, February 29th - March 3rd, 2012, Paris, France*, pages 338–349, 2012. doi:10.4230/LIPIcs.STACS.2012.338.
- 36 George L. Nemhauser and Leslie E. Trotter Jr. Vertex Packings: Structural Properties and Algorithms. *Mathematical Programming*, 8(1):232–248, 1975. doi:10.1007/BF01580444.
- 37 Arfst Nickelsen and Till Tantau. The Complexity of Finding Paths in Graphs with Bounded Independence Number. *SIAM Journal on Computing*, 34(5):1176–1195, 2005. doi:10.1137/S0097539704441642.
- 38 Christos H. Papadimitriou and Mihalis Yannakakis. Optimization, Approximation, and Complexity Classes. *J. Comput. Syst. Sci.*, 43(3):425–440, 1991. doi:10.1016/0022-0000(91)90023-X.
- 39 Michal Pilipczuk, Sebastian Siebertz, and Szymon Torunczyk. Parameterized Circuit Complexity of Model-Checking on Sparse Structures. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2018, Oxford, UK, July 09-12, 2018*, pages 789–798. ACM, 2018. doi:10.1145/3209108.3209136.
- 40 Igor Razgon and Barry O’Sullivan. Almost 2-SAT is Fixed-Parameter Tractable. *Journal of Computer and System Sciences*, 75(8):435–450, 2009. doi:10.1016/j.jcss.2009.04.002.
- 41 Bruce A. Reed, Kaleigh Smith, and Adrian Vetta. Finding Odd Cycle Transversals. *Operations Research Letters*, 32(4):299–301, 2004. doi:10.1016/j.orl.2003.10.009.
- 42 Stefan Szeider. The Parameterized Complexity of k-flip Local Search for SAT and MaxSAT. *Discret. Optim.*, 8(1):139–145, 2011. doi:10.1016/j.disopt.2010.07.003.