On the parameterized complexity of monotone and antimonotone weighted circuit satisfiability[*]

IYAD KANJ[†] DIMITRIOS M. THILIKOS[‡§] GE XIA[¶]

November 7, 2016

Abstract

We consider the weighted monotone and antimonotone satisfiability problems on normalized circuits of depth at most $t \geq 2$, abbreviated WSAT$^+[t]$ and WSAT$^-[t]$, respectively, where the parameter under consideration is the weight of the sought satisfying assignment. These problems model the weighted satisfiability of monotone and antimonotone propositional formulas (including weighted monotone/antimonotone CNF-SAT) in a natural way, and serve as the canonical problems in the definition of the parameterized complexity hierarchy. In particular, WSAT$^+[t]$ ($t \geq 2$) is $\mathsf{W}[t]$-complete for even t and $\mathsf{W}[t-1]$-complete for odd t, and WSAT$^-[t]$ ($t \geq 2$) is $\mathsf{W}[t]$-complete for odd t and $\mathsf{W}[t-1]$-complete for even t. Moreover, several well-studied problems, including important graph problems, can be modeled as WSAT$^+[t]$ and WSAT$^-[t]$ problems in a straightforward manner. We study the parameterized complexity of WSAT$^-[t]$ and WSAT$^+[t]$ with respect to the genus of the circuit. For WSAT$^-[t]$, we give a fixed-parameter tractable (FPT) algorithm when the genus of the circuit is $n^{o(1)}$, where n is the number of the variables in the circuit. For WSAT$^+[2]$ (*i.e.*, weighted monotone CNF-SAT) and WSAT$^+[3]$, which are both $\mathsf{W}[2]$-complete, we also give FPT-algorithms when the genus is $n^{o(1)}$. For WSAT$^+[t]$ where $t \geq 4$, we give FPT-algorithms when the genus is $o(\log^{2/3}(n))$. We also show that both WSAT$^-[t]$ and WSAT$^+[t]$ on circuits of genus $n^{\Omega(1)}$ have the same W-hardness as the general WSAT$^+[t]$ and WSAT$^-[t]$ problem (*i.e.*, with no restriction on the genus), thus drawing a precise map of the parameterized complexity of WSAT$^-[t]$ and of WSAT$^+[2]$ with respect to the genus of the underlying circuit.

As a byproduct of our results, we obtain, via standard parameterized reductions, tight results on the parameterized complexity of several problems with respect to the genus of the underlying graph.

Keywords. Weighted satisfiability; parameterized complexity; genus.

[*]A preliminary version of this paper appeared in proceedings of the *12th International Symposium on Algorithms and Data Structures* (WADS), volume 8037 of *Lecture Notes in Computer Science*, (2013), pages 451-462.

[†]School of Computing, DePaul University, 243 S. Wabash Avenue, Chicago, IL 60604. Email: `ikanj@cs.depaul.edu`.

[‡]Department of Mathematics, National and Kapodistrian University of Athens, Greece and AlGCo project team, CNRS, LIRMM, France. Email: `sedthilk@thilikos.info`.

[§]The third author was co-financed by the E.U. (European Social Fund - ESF) and Greek national funds through the Operational Program "Education and Lifelong Learning" of the National Strategic Reference Framework (NSRF) - Research Funding Program: "Thales. Investing in knowledge society through the European Social Fund".

[¶]Department of Computer Science, Acopian Engineering Center, Lafayette College, Easton PA 18042, USA. Email: `gexia@cs.lafayette.edu`.

1 Introduction

We consider the *weighted satisfiability* problems on monotone and antimonotone normalized circuits of depth at most $t \geq 2$. In the ANTIMONOTONE WEIGHTED SATISFIABILITY problem on normalized circuits of depth at most $t \geq 2$, abbreviated WSAT$^-[t]$, we are given a circuit C of depth t in the *normalized* form [12, 13] (*i.e.*, the output gate is an AND-gate, and the gates alternate between AND-gates and OR-gates) whose input literals are all negative, and an integer parameter $k \geq 0$, and we need to decide if C has a satisfying assignment of weight k (*i.e.*, assigning the value 1 to k variables of C). In the MONOTONE WEIGHTED SATISFIABILITY on normalized circuits of depth at most $t \geq 2$, abbreviated WSAT$^+[t]$, we are given a circuit C of depth t in the normalized form whose input literals are positive, and an integer parameter $k \geq 0$, and we need to decide if C has a satisfying assignment of weight k. Our goal in this paper is to study the parameterized complexity of WSAT$^-[t]$ and WSAT$^+[t]$ with respect to the genus of the circuit. We define the genus of the circuit to be the genus of the underlying undirected graph after the output gate is removed. The reason we exclude the output gate of the circuit in the definition of the genus is two-fold. First, excluding the output gate allows us to use standard FPT-reductions to model problems on graphs satisfying a certain genus upper bound as WSAT$^-[t]$ and WSAT$^+[t]$ problems on circuits that satisfy the same genus upper bound, whereas such modeling would not be possible if the genus is defined to be that of the whole circuit. Second, as it turns out, one obtains the same tight results obtained in the current paper if the genus is defined to be that of the whole circuit. To see this, observe that all positive results (FPT membership results) obtained in this paper carry over because an upper bound on the genus of the whole circuit implies the same upper bound on the genus of the same circuit with the output gate removed. Therefore, showing that the WSAT$^-[t]$ and WSAT$^+[t]$ problems are FPT on circuits whose genus defined with the output gate removed is at most $g(n)$, for some function $g(n)$, where n is the number of variables in the circuit, will imply that the WSAT$^-[t]$ and WSAT$^+[t]$ problems are FPT on circuits whose genus defined with the output gate included is at most $g(n)$. On the other hand, the same straightforward (padding) arguments used to obtain the W-hardness results in this paper (Lemmas 3.5 and 4.2) with the genus defined after removing the output gate, can be used to obtain W-hardness results for the same upper bound on the genus when the genus is defined to be that of the whole circuit. We mention that the WEIGHTED CIRCUIT SATISFIABILITY problem on depth-t planar circuits with the output gate included is solvable in polynomial time [7], whereas it can be easily shown that WSAT$^-[t]$ and WSAT$^+[t]$ are NP-complete on planar circuits (and hence on circuits of any genus) with the output gate removed. We also note that WEIGHTED CIRCUIT SATISFIABILITY on planar circuits with unbounded depth is known to be W$[P]$-complete [1]. Recently, in [23], Marx proved that weighted monotone/antimonotone circuit satisfiability has no FPT-approximation algorithm with any approximation ratio function ρ, unless W[1] = FPT.

1.1 Motivation and related work

The problems under consideration are of prime interest both theoretically and practically. From the theoretical perspective, they naturally represent the weighted satisfiability of (montone/antimontone) t-normalized propositional formulas, *i.e.*, products-of-sums-of-products... (see [12,13]), including the canonical problems weighted antimonotone/monotone CNF-SAT. Moreover, the WSAT$^-[t]$ and the WSAT$^+[t]$ problems are the canonical complete problems for the different levels of the parameterized complexity hierarchy — the W-hierarchy, and the W-hierarchy can be defined based on them [12, 13]. Therefore, determining the underlying structure that makes these problems (parameterized) tractable is important from the perspective of complexity theory. From a more practical perspective, WSAT$^{-}[t]$ and WSAT$^{+}[t]$ can model several natural problems. Therefore, parameterized algorithms for WSAT$^{-}[t]$ and WSAT$^{+}[t]$ can be used to obtain parameterized algorithms for some natural problems via FTP-reductions to/from WSAT$^{-}[t]$ and WSAT$^{+}[t]$, as we shall see in Section 5.

The computational complexity of many natural problems on planar graphs, and more generally on graphs whose genus meets certain upper bounds, have been extensively researched (see [4, 9, 10, 15, 16], among others). In particular, it was shown that the bounded-genus property plays a key-role in determining the computational complexity (parameterized complexity including kernelization, subexponential-time computability, and approximation) of a large class of graph problems. For example, using *bidimensionality theory*, it was shown in [9] that a large class of graph problems admit subexponential-time parameterized algorithm on graphs whose genus is upper bounded by a constant. For graphs of larger genus (could be unbounded), it was shown in [8] that the genus characterizes the computational complexity (parameterized complexity, approximation, and subexponential-time computability) of some natural graph problems, including INDEPENDENT SET and DOMINATING SET. We note that the techniques used in [8] to characterize the parameterized complexity of certain graph problems with respect to the genus of the graph are problem specific, and are not applicable to the weighted satisfiability problems under consideration in this paper.

Research results on planar circuits, and on satisfiability problems defined on certain structures that are planar or that satisfy certain structural properties, are abundant. Planar Boolean circuits have been extensively studied in the literature as they can be used to study VLSI chips, and they play an important role in deriving computational lower bounds for Boolean circuits [24, 26, 28]. After Lipton and Tarjan established their celebrated planar separator theorem, one of the first applications of the separator theorem they gave, was to derive lower bounds on the size of Boolean circuits that compute certain important functions [22]. The computational power of monotone planar circuits were also considered (*e.g.*, see [2, 21]). Khanna and Motwani [19] studied the approximation of instances of satisfiability problems (weighted and unweighted) whose underlying structure is planar. More specifically, they studied satisfiability problems defined based on disjunctive normal form (DNF) formulas. The incidence graph of an instance of such problems is a simple bipartite graph that has a vertex for each variable and a vertex for each formula, and an edge between them if the variable occurs in the formula. They derived polynomial-time approximations schemes for instances of these problems whose underlying incidence graph is planar [19]. Cai et al. [6] studied the parameterized complexity of the satisfiability problems introduced by Khanna and Motwani [19], and showed that these problems are W[1]-hard even when the underlying incidence graph is planar. Researchers have also studied the parameterized complexity of CNF-SAT with respect to the treewidth of a graph defined by the corresponding formula (for example, see [25]).

1.2 Our results and techniques

We obtain the following results regarding WSAT$^{-}[t]$ ($t \geq 2$), which is W$[t]$-complete for odd t and W$[t-1]$-complete for even t [12, 13] (in what follows, n is the number of variables in the circuit):

(i) **Tight results**: We give an FPT-algorithm for WSAT$^{-}[t]$ when the genus is $n^{o(1)}$. Moreover, we show that WSAT$^{-}[t]$ has the same W-hardness status as the general WSAT$^{-}[t]$ problem when the genus is $n^{\Omega(1)}$.

(ii) **Applications**: We show that INDEPENDENT SET ON HYPERGRAPHS and the RED-BLUE NONBLOCKER problems are in FPT on (hyper)graphs of genus $N^{o(1)}$ and W[1]-complete on (hyper)graphs of genus $N^{\Omega(1)}$ (N is the number of red vertices in RED-BLUE NONBLOCKER and the total number of vertices in INDEPENDENT SET ON HYPERGRAPHS).

The techniques used for deriving the FPT results in (i) can be summarized as follows. We first show that in FPT-*time* we can reduce an instance of WSAT$^-[t]$ on a circuit of genus $n^{o(1)}$ to an equivalent instance of WSAT$^-[t]$ in which the number of occurrences of the literals is linear in n; we bound the number of occurrences using combinatorial arguments that are based on Euler-type results for (multi)hypergraphs whose genus meets certain upper bounds (Proposition 3.3). We then show that any instance of WSAT$^-[t]$ with a linear number of occurrences (and no zero-variables) admits a satisfying assignment whose weight is lower bounded by a function of n (Theorem 3.4). Combining the preceding two results, we obtain the FPT results in (i) (Theorem 3.6). The hardness result in (i) is derived by a simple FPT-reduction from the general WSAT$^-[t]$ problem (Theorem 3.6). The results in (ii) about INDEPENDENT SET ON HYPERGRAPHS and RED-BLUE NONBLOCKER are derived using standard FPT-reductions (Theorem 5.2).

We obtain the following results regarding WSAT$^+[t]$ ($t \geq 2$), which is known to be W$[t]$-complete for even t and W$[t-1]$-complete for odd t [12,13]:

(1) **Tights results for** $t = 2$ **and** $t = 3$: We give FPT-algorithms for WSAT$^+[2]$ (*i.e.*, weighted monotone CNF-SAT) and WSAT$^+[3]$ when the genus is $n^{o(1)}$ and show that the problems are W[2]-complete when the genus is $n^{\Omega(1)}$.

(2) **Results for** $t \geq 4$: We give an FPT-algorithm for WSAT$^+[t]$ when the genus is $o(\log^{2/3}(n))$, and show that WSAT$^+[t]$ has the same W-hardness as the general WSAT$^+[t]$ problem when the genus is $n^{\Omega(1)}$.

(3) **Applications**: We show that RED-BLUE DOMINATING SET, HITTING SET, and SET COVER are FPT if the underlying graph/hypergraph has genus $N^{o(1)}$ and W[2]-complete if the underlying graph/hypergraph has genus $N^{\Omega(1)}$ (N is the number of red vertices in RED-BLUE DOMINATING SET, the cardinality of the vertex-set in HITTING SET, and the number of sets in SET COVER).

All FPT results in (1) and (2) rely on a result (Proposition 4.1) showing that, for circuits of genus $n^{o(1)}$, there is a *Turing*-FPT self-reduction that reduces an instance of WSAT$^+[t]$ to FPT-*many* instances of WSAT$^+[t]$ in which the number of gates that are incoming to the output gate of the circuit is a function of the parameter. Using this result, we can directly show that WSAT$^+[2]$ is FPT when the genus is $n^{o(1)}$ (Theorem 4.3). For $t > 2$, we show that the aforementioned result implies that the treewidth of the resulting circuit is $o(\log n) \cdot \sqrt{k}$ if its genus is $o(\log^{2/3}(n))$; this allows us to apply an intricate dynamic programming approach to show that the problem on genus $o(\log^{2/3}(n))$ circuits is FPT (Theorem 4.8). For $t = 3$, we show that the aforementioned dynamic programming approach can be modified to run in fpt-time even when the genus of the circuit is $n^{o(1)}$ (Theorem 4.9). The hardness results for WSAT$^+[t]$ on circuits of genus $n^{\Omega(1)}$ in (1) and (2) are derived by simple FPT-reductions from the general WSAT$^+[t]$ problem (Lemma 4.2). The results in (3) can be derived using standard FPT-reductions (Theorem 5.1).

Remark 1.1. None of the algorithms presented in the current paper needs to know in advance, nor needs to decide, whether or not the minimum genus of the input circuit satisfies the required upper bounds. More specifically, if the input instance of the problem satisfies the required genus upper bound, then the algorithm will solve the instance correctly in the specified running time.

On the other hand, if the input instance does not satisfy the required genus upper bound, then the algorithm either rejects the instance because it does not satisfy the genus upper bound requirement, or solves it correctly, (both) in the specified running time. Therefore, one does not need to limit the algorithms in the current paper to the input instances in which the genus is guaranteed to satisfy the required given upper bounds (*i.e.*, a witness is unnecessary).

2 Preliminaries

2.1 Graphs, hypergraphs, and genus

We assume familiarity with the basic terminology and definitions in graph theory and parameterized complexity, and refer the reader to [12, 13, 29].

All graphs in this paper are loop-less, but may have multiple edges. A graph without multiple edges is a *simple* graph. Let G be an undirected graph. For an edge $e = uv$ in G, *contracting* e means removing the two vertices u and v from G, replacing them with a new vertex w, and for every vertex y in the neighborhood of v or u in G, adding in the new graph an edge wy whose multiplicity is the sum of the multiplicities of the edges of G between v and y and between u and y. If in the above definition we do not sum up multiplicities, and if the initial graph G is a simple graph, then we call the operation *simple contraction*, or for short *s-contraction*.

Given a vertex-set $S \subseteq V(G)$ such that the subgraph of G induced by S, denoted $G[S]$, is connected, *contracting* S means contracting the edges between the vertices in S to obtain a single vertex at the end.

A *hypergraph* $\mathcal{H} = (V, E)$ consists of a *vertex set* $V = V(\mathcal{H})$ and an *edge set* $E = E(\mathcal{H})$ so that $e \subseteq V$ for every $e \in E$. If E is allowed to be a multiset we call $\mathcal{H}$ a *multihypergraph*. We also call the edges in a hypergraph *hyperedges*.

A graph has *genus* at most g if it can be drawn on a surface of genus g (a sphere with g handles) without edge intersections. We say a (multi)hypergraph $\mathcal{H}$ is *embeddable in a surface* if the bipartite incidence graph obtained from $\mathcal{H}$ by replacing each of its hyperedges by a vertex adjacent to all the vertices in the hyperedge is embeddable in that surface. In particular, this definition allows us to speak of *(multi)hypergraph of genus g*. We refer the reader to [17] for more information on the genus of a graph. The following lemmas will be useful:

Lemma 2.1 ([18]). *A multihypergraph of genus at most g on n vertices has at most $2n + 4g - 4$ hyperedges containing at least three vertices, unless $n = 1$ and $g = 0$.*

Lemma 2.2 (Euler). *A simple graph of genus g on n vertices contains at most $3n + 6g - 6$ edges if $n \geq 3$.*

Lemma 2.3 ([18]). *A hypergraph of genus at most g on n vertices has at most $6n + 10g - 10$ hyperedges if $n \geq 3$.*

2.2 Circuits, weighted satisfiability, and complexity functions

A *circuit* is a directed acyclic graph. The vertices of indegree 0 are called the (input) *variables*, and are labeled either by *positive literals* x_i or by *negative literals* $\overline{x}_i$. The vertices of indegree larger than 0 are called the *gates* and are labeled with Boolean operators AND or OR. A special gate of outdegree 0 is designated as the *output* gate. We do not allow NOT gates in the above circuit model, since by De Morgan's laws, a general circuit can be effectively converted into the above circuit model. A circuit is said to be *monotone* (resp. *antimonotone*) if all its input literals are positive (resp. negative). The *depth* of a circuit is the maximum distance from an input variable to the output gate of the circuit. A circuit represents a Boolean function in a natural way. The size of a circuit C, denoted $|C|$, is the size of the underlying graph (*i.e.*, number of vertices and edges). An *occurrence* of a literal in C is an edge from the literal to a gate in C. Therefore, the total number of occurrences of the literals in C is the number of outgoing edges from the literals in C to its gates. The *genus of a circuit* is the genus of the underlying undirected graph after the output gate has been removed (as discussed in Section 1 — see Figure 1 for illustration). We will abuse the notation and say that a circuit has genus g if it has genus at most g. The justification of the preceding statement can be found in Remark 2.5 at the end of this section.

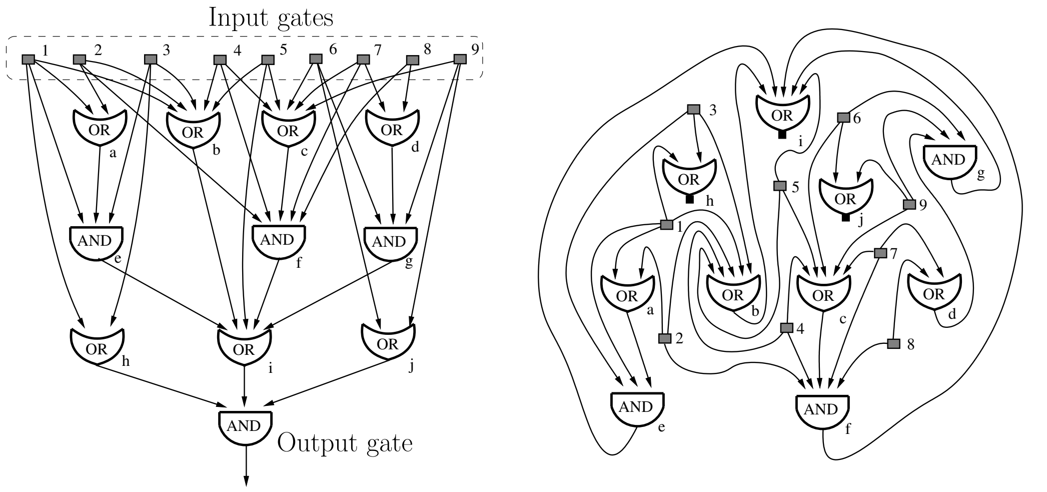

Figure 1: A planar monotone circuit in normalized form of depth 4 together with a plane embedding of it (without the output gate).

We consider circuits whose output gate is an AND-gate and that are in the *normalized* form (see [12,13]). In the normalized form every (nonvariable) gate has outdegree at most 1, and above the output AND-gate, the gates are structured into alternating levels of ORs-of-ANDs-of-ORs... (we visualize the circuit in an hierarchical fashion where the output gate is the lowest one). We denote a circuit that is in the normalized form and that is of depth at most $t \geq 2$ by a Π_t circuit. We write Π_t^+ to denote a monotone Π_t circuit, and Π_t^- to denote an antimonotone Π_t circuit. We do not assume that the literals appear at the same (top) level of the circuit (notice that the circuit in Figure 1 has a planar embedding where the literals are not drawn in the top).

Throughout the paper, we implicitly assume that the following **simplification rules** hold: every gate with outdegree 0 except the output gate is removed, every gate has indegree at least 2, and no two gates of the same type such that one is incoming to the other exist.

We say that a truth assignment τ to the variables of a circuit C *satisfies* a gate g in C if τ makes the gate g have value 1, and that τ *satisfies the circuit* C if τ satisfies the output gate of C. A circuit C is *satisfiable* if there is a truth assignment to the input variables of C that satisfies C. The *weight* of an assignment τ is the number of variables assigned value 1 by τ. An indegree-2 gate is called a *2-literal gate* if both its incoming edges are from literals. A *critical gate* in a Π_t circuit C is an OR-gate that is connected to the output AND-gate of the circuit; clearly, any satisfying assignment to C must satisfy all critical gates in C. If we remove the literals from C, we obtain a directed graph whose underlying undirected graph is a tree T_C. If we root T_C at the output gate of C, we can now use the terms *child(ren)*, *parent*, *grandparent* of a gate in T_C in a natural way. Note that every literal in C is connected to some gates in T_C. For a gate g in T_C, we denote by T_g the subtree of T_C rooted at g. We may regard an edge in T_C between a child g' of a gate g and g, or between a literal and gate g, as an incoming edge to g.

A *parameterized problem* Q is a subset of $\Omega^* \times \mathbb{N}$, where Ω is a fixed alphabet and $\mathbb{N}$ is the set of all non-negative integers. Each instance of the parameterized problem Q is a pair (x, k), where the second component, i.e., the non-negative integer k, is called the *parameter*. We say that the parameterized problem Q is *fixed-parameter tractable* [12,13], shortly FPT, if there is a (parameterized) algorithm, also called FPT-*algorithm*, that decides whether an input (x, k) is a member of Q in time $f(k) \cdot |x|^{O(1)}$, where f is a computable function. Let FPT denote the class of all fixed-parameter tractable parameterized problems. (We abused the notation "FPT" above for simplicity.)

A parameterized problem Q is FPT-*reducible* to a parameterized problem Q' if there is an algorithm that transforms each instance (x, k) of Q into an instance (x', k') of Q' in time $f(k) \cdot |x|^{O(1)}$, such that $k' \leq g(k)$ and $(x, k) \in Q$ if and only if $(x', k') \in Q'$, where f and g are computable functions.

Based on the notion of FPT-reducibility, a hierarchy of fixed-parameter intractability, *the* W-*hierarchy* $\bigcup_{t \geq 0} \mathsf{W}[t]$, where $\mathsf{W}[t] \subseteq \mathsf{W}[t+1]$ for all $t \geq 0$, has been introduced, in which the 0-th level $\mathsf{W}[0]$ is the class FPT. The hardness and completeness have been defined for each level $\mathsf{W}[i]$ of the W-hierarchy for $i \geq 1$ [12,13]. It is commonly believed that $\mathsf{W}[1] \neq \mathsf{FPT}$ (see [12,13]). The $\mathsf{W}[1]$-hardness has served as the main working hypothesis of fixed-parameter intractability.

For $t \geq 2$, the WEIGHTED Π_t-CIRCUIT SATISFIABILITY problem, abbreviated WSAT$[t]$ is for a given Π_t-circuit C and a given parameter k, to decide if C has a satisfying assignment of weight k. The WEIGHTED MONOTONE Π_t-CIRCUIT SATISFIABILITY problem, abbreviated WSAT$^+[t]$, and the WEIGHTED ANTIMONOTONE Π_t-CIRCUIT SATISFIABILITY problem, abbreviated WSAT$^-[t]$ are the WSAT$[t]$ problems on monotone circuits and antimonotone circuits, respectively. We denote by WSAT$^-$ the WSAT$^-[2]$ problem, and by WSAT$^+$ the WSAT$^+[2]$ problem (*i.e.*, the weighted antimonotone/monotone CNF-SAT problem). It is known that for each integer $t \geq 2$: WSAT$^+[t]$ is $\mathsf{W}[t]$-complete for even t and $\mathsf{W}[t-1]$-complete for odd t, and WSAT$^-[t]$ is $\mathsf{W}[t]$-complete for odd t and $\mathsf{W}[t-1]$-complete for even t [12,13].

The (time) complexity functions used in this paper are assumed to be proper complexity functions that are unbounded and nondecreasing. For a complexity function $f : \mathbb{N} \to \mathbb{N}$, we define its inverse, f^{-1}, by $f^{-1}(h) = \max\{q \mid f(q) \leq h\}$ if $\{q \mid f(q) \leq h\} \neq \emptyset$, and $f^{-1}(h) = 0$ otherwise. Since the function f is nondecreasing and unbounded, the function f^{-1} is also nondecreasing and unbounded, and satisfies $f(f^{-1}(h)) \leq h$. We shall also assume that the complexity functions and their inverses can be computed efficiently (i.e., in time linear in the input size and the value of the function). The $o(\cdot)$ notation used in this paper denotes the $o^{\text{eff}}(\cdot)$ notation (see, for instance, [13]). More formally, for any two computable functions $f, g : \mathbb{N} \to \mathbb{N}$, by writing $f(n) = o(g(n))$ we mean that there exists a computable nondecreasing unbounded function $\mu(n) : \mathbb{N} \to \mathbb{N}$, and $n_0 \in \mathbb{N}$, such that $f(n) \leq g(n)/\mu(n)$ for all $n \geq n_0$.

The following lemma is useful.

Lemma 2.4. *For any complexity function h, and any function g that belongs to either of the two families of functions $O(N^{o(1)h(k)})$ or $O((\log N)^{h(k)})$, there exists a complexity function f such that g is in $f(k)N^{O(1)}$. Therefore, if a parameterized problem is solvable in time that is upper bounded by $g(N, k)$ (where N is the input length and k is the parameter), then the parameterized problem is solvable in* FPT*-time, and hence belongs in* FPT*.*

Proof. Let h be a complexity function, and suppose that the parameterized problem is solvable in time $g(N, k) \in O(N^{h(k)/\mu(N)})$, for some complexity function $\mu(N)$. If $h(k) \leq \mu(N)$ then

$O(N^{h(k)/\mu(N)}) = O(N)$. On the other hand, if $h(k) > \mu(N)$ then $N < \mu^{-1}(h(k))$, and if we let $f(k) = \mu^{-1}(h(k))^{h(k)}$, we have $N^{h(k)/\mu(N)} \leq \mu^{-1}(h(k))^{h(k)/\mu(N)} \leq f(k)^{1/\mu(N)} \leq f(k)$. It follows that in both cases $O(N^{h(k)/\mu(N)}) \leq f(k)N^{O(1)}$. The proof when $g(N,k) \in O((\log N)^{h(k)})$ follows the same method by distinguishing the two cases $h(k) \leq \log(\log N)$ and $h(k) > \log(\log N)$. □

Remark 2.5. When we consider WSAT$^+[t]$ (resp. WSAT$^-[t]$) on circuits of genus $g(n)$, for some function $g(n)$ of the number of variables n of the circuit, we mean WSAT$^+[t]$ (resp. WSAT$^-[t]$) restricted to all the instances in which the genus of the underlying circuit is *upper bounded* by the prespecified function $g(n)$. This set of instances includes the instances in which the genus of the underlying circuit is smaller than $g(n)$. Note that this does not affect any of the results in the paper, as the FPT results hold as well for the instances of smaller genus (than $g(n)$) in the set of instances under consideration, and the hardness results hold due to the hardness of the restriction of the set of instances under consideration to those instances whose genus is $g(n)$.

3 The antimonotone case

In this section we give tight results on the parameterized complexity of the WSAT$^-[t]$ problem, where $t \geq 2$ is an integer, with respect to the genus of the circuit. We start with the following definition.

Definition 3.1. Let C be a Π_t^- circuit, and let x_i be a variable in C. We say that x_i is a *zero-variable* for C if assigning $x_i = 1$ causes C to evaluate to 0. Therefore, any zero-variable for C must be assigned the Boolean value 0 in a satisfying truth assignment for C. A *nonzero-variable* for C is a variable that is not a zero-variable for C. A Π_t^- circuit C *has no zero-variables* if all the variables in C are nonzero-variables.

We note that determining whether or not a variable x_i is a zero-variable for a Π_t^- circuit C can be done in polynomial time.

Proposition 3.2. *Let (C,k) be an instance of* WSAT$^-[t]$ *($t \geq 2$) such that the genus of C is $g(n) = n^{o(1)}$. In* FPT*-time, we can either decide (C,k), or we can reduce (C,k) to an equivalent instance (C',k) where C' has genus at most $g(n)$ and no zero-variables, and such that the number of variables n' in C' satisfies $g(n) \leq n' \leq n$.*

Proof. Observe that if (C,k) has a satisfying assignment of weight k, then none of the variables assigned 1 by such an assignment can be a zero-variable of C.

Suppose first that the number of nonzero-variables in C is smaller than $g(n) = n^{o(1)}$, and let $\mathcal{N}$ be the set of nonzero-variables of C. We enumerate each subset S of $\mathcal{N}$ of size k as a candidate subset of variables that will be assigned 1 by a satisfying assignment of weight k for C. For each such candidate subset S, we assign the variables in S the value 1 and the remaining variables in C the value 0, and check if the assignment satisfies C; if it does, we accept (C,k). If no enumerated subset leads to acceptance, we reject (C,k). The number of the enumerated subsets is $\binom{|\mathcal{N}|}{k} = n^{o(1)k}$. By Lemma 2.4, the above algorithm runs in FPT-time.

We may now assume that the number of nonzero-variables in C, n', is at least $g(n) = n^{o(1)}$. Let C' be the circuit obtained from C by assigning the zero-variables of C the value 0. Observe that this assignment does not introduce zero-variables, and hence the resulting circuit C' has no zero-variables, and satisfies the statement of the proposition. □

Next, we need a structural result showing that any Π_t^- circuit whose genus is at most linear (in the number of variables) can be reduced to an equivalent one on the same set of variables whose size is also linear. We first need the following definition and reductions.

Let v and v' be vertices in a Π_t^- circuit C. We say that v and v' are *equivalent* if v and v' correspond to the same literal, or if both v and v' are 2-literal gates that are of the same type (either both are AND-gates or both are OR-gates) and have the same two literals incoming to them.

We apply the following reduction rule repeatedly until it is not applicable:

Reduction Rule 3.1. Let C be a Π_t^- circuit, and let g be a gate in C. Let v be a literal or a 2-literal gate that is incoming to g. Do the following:

(a) If there exists a vertex $v' \neq v$ that is equivalent to v, such that v' is incoming to g, then let C' be the circuit resulting from C after removing the edge from v' to g.

(b) If g is an OR-gate and there exists a gate $g' \neq g$ in the subtree T_g of T_C and a vertex v' equivalent to v such that v' is incoming to g', then let C' be the circuit resulting from C after performing the following: if g' is an AND-gate then remove g', and if g' is an OR-gate then remove the edge from v' to g'.

(c) If g is an AND-gate and there exists a gate $g' \neq g$ in T_g and a vertex v' equivalent to v such that v' is incoming to g', then let C' be the circuit resulting from C after performing the following: if g' is an OR-gate then remove g', and if g' is an AND-gate then remove the edge from v' to g'.

Then the resulting circuit C' is a Π_t^- circuit that is equivalent to C.

Proof. We prove the correctness for the case when v is a literal. The proof is very similar for the case when v is a 2-literal gate.

It suffices to show that any truth assignment τ satisfies C if and only if it satisfies C'. Since the only differences between C and C' occur in T_g (including the literals connected to the gates in T_g), it suffices to show that the value of g induced by τ in C is the same as that in C'. This is clear for part (a), so we prove it for part (b), and the proof for (c) is similar. Note that, by the simplification rules, we can assume that every gate has indegree at least 2.

If v is assigned 1 by τ, then clearly the value of g induced by τ in both C and C' is 1, and hence is the same. Suppose now that v is assigned 0 by τ. An AND-gate in T_g that v is incoming to evaluates to 0 by τ, and hence its removal from C does not affect the value of g induced by τ; similarly, the value of an OR-gate in T_g to which v is incoming, is not affected by the removal of the connection from v to this gate, and hence this removal does not affect the value of g induced by τ. It follows that the value of g induced by τ is the same in both C and C'. ☐

Note that all the simplification rules and the reduction rule do not increase the genus of C, nor do they decrease the number of variables/literals in C. Moreover, these operations can be carried out in time polynomial in the size of the circuit.

Proposition 3.3. *Let C be a Π_t^- circuit on n variables of genus $g(n) \leq n$. In polynomial time we can reduce C to an equivalent Π_t^- circuit C' of genus $g(n)$ on the same set of variables such that the size of C' is $O(n)$.*

Proof. We prove only the upper bound on the number of occurrences. The upper bound on the size follows from the upper bound on the number of occurrences, the fact that each gate in C has outdegree 1, and the fact that C has constant depth.

We apply Reduction Rule 3.1 to C until this is no longer applicable. (We also assume that the simplification rules are applied as discussed before.) Let C' be the resulting circuit. From the above reduction rule, we know that C' is equivalent to C. Since none of these rules remove any literals, C' has the same variables as C. Moreover, all operations performed by the reduction and simplification rules either remove edges, gates, or are edge contractions. Therefore, the genus of C' is at most $g(n)$. It remains to show that the number of occurrences of the literals in C' is $O(n)$.

By part (a) of Reduction Rule 3.1, we know that the number of literals incoming to the output gate of C is $O(n)$. Let C'^{-} be C' with the output gate removed. It suffices to show that the number of occurrences in C'^{-} is $O(n)$. To simplify the counting, we divide the occurrences of the literals in C'^{-} into three types: (1) occurrences of literals incoming to a gate g' such that g' has indegree at least 3 and all incoming edges to g' are from literals; (2) occurrences of literals incoming to 2-literal gates; and (3) all other occurrences, which are the occurrences of literals incoming to a gate that has at least one gate incoming to it. Next, we upper bound the number of occurrences of each type. (In Figure 1, with the output gate removed, the type-(1) occurrences are the edges $(1,b),(2,b),(3,b),(4,b),(5,b),(4,c),(5,c),(6,c),(7,c),(9,c)$; the type-(2) occurrences are $(1,a),(2,a),(7,d),(8,d),(1,h),(3,h),(6,j),(9,j)$; and all other occurences are type-(3) occurrences.)

To upper bound the number of type-(1) occurrences, we define the multihypergraph $\mathcal{H}$ whose vertex-set is the set of literals in C'. Call a gate g' of degree at least 3 whose incoming edges are all from literals, a *type-(1) gate*. (In Figure 1, the type-(1) gates are the two gates labeled b and c.) For each type-(1) gate g', we correspond a hyperedge in $\mathcal{H}$ that contains the literals that are incoming to g'. Clearly, the number of occurrences of the literals that are incoming to the type-(1) gates is the same as the total number of occurrences of the vertices of $\mathcal{H}$ in its hyperedges. Since the genus of C'^{-} is at most $g(n)$, by the definition of the genus of a hypergraph, the genus of $\mathcal{H}$ is at most $g(n)$ since its incidence graph is a subgraph of the underlying graph of C'^{-}. Since each hyperedge in $\mathcal{H}$ has size at least 3, by Lemma 2.1, the number of hyperedges in the multihypergraph $\mathcal{H}$ is $O(n+g(n)) = O(n)$. Therefore, the incidence graph $\mathcal{I}$ of $\mathcal{H}$ has $O(n)$ vertices and genus $g(n)$. By Lemma 2.2, the number of edges in $\mathcal{I}$, which is the same as the total number of vertices in the hyperedges of $\mathcal{H}$, is $O(n)$. This shows that the number of type-(1) occurrences is $O(n)$.

To upper bound the number of type-(2) occurrences, we upper bound the number of 2-literal gates. First, consider the set $\mathcal{G}_0$ of 2-literal gates that are incoming to the output gate of C', and ignore all other gates for now. (In Figure 1, the 2-literal gates are the gates labeled a, d, h, j, and only gates h and j are incoming to the output gate.) We start by upper bounding the cardinality of $\mathcal{G}_0$. Since all gates in $\mathcal{G}_0$ are incoming to the output gate of C', by Reduction Rule 3.1, and since all gates in $\mathcal{G}_0$ are OR-gates, any pair of literals in C can be incoming to at most one gate in $\mathcal{G}_0$. Therefore, we can define a (simple) graph whose vertex-set is the set of literals in C', and whose edges correspond to the gates in $\mathcal{G}_0$. Clearly, the genus of the constructed graph is $g(n)$. By Lemma 2.2, the number of edges in this graph, which is the same as the number of gates in $\mathcal{G}_0$, is $O(n)$. It follows that the cardinality of $\mathcal{G}_0$ is $O(n)$, and hence, the number of type-(2) occurrences that are incoming to gates in $\mathcal{G}_0$ is $O(n)$.

Now we upper bound the number of 2-literal gates that are not in $\mathcal{G}_0$; let $\mathcal{G}_1$ be the set of these gates. First, we upper bound the number of critical gates in C' that are not in $\mathcal{G}_0$ by $O(n)$. To do so, observe that each such critical gate g' has at least three literals incoming to the gates in $T_{g'}$ (note that there are no gates of indegree 1). By contracting the edges in $T_{g'}$, for each critical gate g', and removing any resulting multiple edges, we obtain a vertex that is connected to at least three distinct literals in C'; the fact that the resulting vertex is connected to at least three distinct literals follows from the simplification rules and from Reduction Rule 3.1, and can be easily verified by the reader. We correspond this resulting vertex with gate g'. Now by defining a multihypergraph whose vertices are the literals in C', and whose hyperedges correspond to the vertices resulting from the contractions, we can upper bound the number of such vertices, and hence the number of critical gates in C' by $O(n)$, in a similar fashion to that of bounding the type-(1) gates above. (Note that the genus of the defined multihypergraph is at most $g(n)$ since its incidence graph is a subgraph of a contraction of C'^{-}.) To upper bound the number of gates in $\mathcal{G}_1$, apply the following operation until it is no longer applicable: For each gate g' in $\mathcal{G}_1$, if g' is not incoming to a critical gate, contract the edge between the parent of g' in $T_{C'}$ and the grandparent of g' in $T_{C'}$. After the application of the aforementioned operation, each gate in $\mathcal{G}_1$ is incoming to a critical gate, and has exactly two literals incoming to it. Now define a multihypergraph whose vertex-set consists of the set of literals in C' plus the critical gates, and whose hyperedges contain the vertices that the gates in $\mathcal{G}_1$ are adjacent to after these contractions; note that each hyperedge in this multihypergraph has size at least 3. Clearly, the defined multihypergraph has genus $g(n)$ since it is a contraction of a subgraph of C'^{-}. Since the number of critical gates in C' is $O(n)$, it follows from Lemma 2.1 that the number of gates in $\mathcal{G}_1$ is $O(n)$. Summing up, the number of 2-literal gates in C' is $O(n)$, and hence the number of type-(2) occurrences is $O(n)$.

Finally, to upper bound the type-(3) occurrences, we again define a multihypergraph $\mathcal{H}$ of genus $g(n)$ whose vertex-set is the set of literals in C', and use a charging scheme to charge the type-(3) occurrences to the total number of occurrences of the vertices of $\mathcal{H}$ in its hyperedges. To ensure that the genus of $\mathcal{H}$ is $g(n)$, we rely on the forest $\mathcal{F}$ in C'^{-}, resulting from $T_{C'}$ after removing the output gate of C', when defining $\mathcal{H}$. Call a gate a *type-(3) gate* if it has a type-(3) occurrence incoming to it. (In Figure 1, the type-(3) gates are the gates labeled e, f, g, i.) We define the *level* of a gate to be the distance from it to the output gate of C'. We start the charging argument at the type-(3) gates at the highest level of the circuit, and go from the top to the bottom (we assume that the output gate is at the bottom of the circuit). Since C' has no zero-variables, no type-(3) occurrence is incoming to the output gate of C', and hence this charging scheme will stop at the critical gates of C'. Consider a type-(3) gate g' at the highest level. Since g' is not a type-(2) gate and its indegree is more than 1, the number of distinct literals incoming to the subtree $T_{g'}$ in $\mathcal{F}$ is at least 3. Note that any literal that is incoming to g' is not incoming to any other gate in $T_{g'}$ by Reduction Rule 3.1. Therefore, by contracting $T_{g'}$ to a single vertex and removing any resulting multiple edges, we get a vertex that is adjacent to all the literals that are incoming to $T_{g'}$, including the type-(3) literals (*i.e.*, the literals in the type-(3) occurrences) incoming to g', and such that the degree of this vertex is at least 3. We associate a hyperedge in $\mathcal{H}$ with the vertex resulting from this contraction that contains the literals incoming to the resulting vertex. Note that each type-(3) occurrence that is incoming to g' corresponds to a literal contained in the created hyperedge. In particular, since each type-(3) literal incoming to g' is not incoming to any other gate in $T_{g'}$, no multiple edge that was removed corresponds to any such literal, and all such literals incoming to g' are accounted for (*i.e.*, charged to) by the corresponding literals in the defined hyperedge. Consider now a type-(3) gate g', and assume inductively, that we finished processing all type-(3) gates above it. We can assume that g' has at least one type-(3) gate above it; otherwise, the treatment is similar to that of the base case. If more than one type-(3) gate in $T_{g'}$ have been charged in the above scheme, we keep one of them, and remove the edges between each other gate and its parent in $T_{g'}$, thus disconnecting the (contracted) vertex corresponding to the gate from $\mathcal{F}$; after this process, exactly one type-(3) gate in the resulting $T_{g'}$ was charged earlier in the charging scheme. Again, note that no type-(3) literal incoming to g' can be incoming to any gate in $T_{g'}$. Now we contract the edges in $T_{g'}$ and remove any resulting multiple edges to form a hyperedge of size at least 3 that contains all type-(3) occurrences incoming to g' (this can be viewed as if we are adding the type-(3) literals incoming to g' to the hyperedge corresponding to the single charged type-(3) gate in $T_{g'}$). This charging scheme stops at the critical gates of C'. At that point, we have defined a multihypergraph $\mathcal{H}$ whose genus is $g(n)$ since all the hyperedges in $\mathcal{H}$ were defined based on contractions of subtrees in $\mathcal{F}$. The total number of type-(3) occurrences in C' is at most the total number of occurrences of the vertices of $\mathcal{H}$ in its hyperedges. Using a similar argument to that used for upper bounding the number of type-(1) occurrences, we conclude that the number of type-(3) occurrences in C' is $O(n)$.

It follows that the total number of occurrences of the literals in C' is $O(n)$. This completes the proof. $\square$

The following theorem shows that a Π_t^- circuit with no zero-variables and with a linear number of (literal) occurrences can always be satisfied with an (increasing) function of n variables assigned 1. This result is of independent interest, as we shall see in Section 5.

Theorem 3.4. *Let C be Π_t^- circuit with n variables such that C has no zero-variables and the number of occurrences of the literals in C is $O(n)$. C has a satisfying assignment in which at least $f(n) = \log^{(d^t)} n$ variables are assigned 1, where $\log^{(i)}$ indicates the logarithm (base 2) applied i times, and $d > 0$ is an integer constant that depends on the upper bound on the number of occurrences.*[1]

Proof. We will prove the statement of the theorem by induction on t. We will prove the statement of the theorem for the case in which the degree of every variable (or equivalently literal) in C is upper bounded by some integer constant $d > 0$; the statements is true for any constant upper bound d on the maximum degree of the variables. If the degree of every variable in C is not upper bounded by some constant $d > 0$, then since the number of occurrences of the literals is $\leq d'n$, for some constant d', there is a linear number of variables $n' \geq n/c'$, for some constant $c' \geq 1$, in C each of which is of degree at most d'. By assigning 0 to the variables of degree larger than d' (satisfied gates are then removed), we obtain a circuit C' with n' variables, each of which is of degree at most d'. Because we will prove the statement of the theorem when the degree of each variable is upper bounded by a constant, and for any such constant upper bound, this will imply that C', and hence C, has a satisfying assignment in which at least $\log^{(d'^t)}(n') \geq \log^{(d'^t)}(n/c') \geq \log^{(d'^t)}(n) - c''$ variables are assigned 1, for some constant $c'' > 0$. By choosing an integer constant $d > d'$ large enough, we conclude that C has a satisfying assignment in which at least $\log^{(d^t)}(n)$ variables are assigned 1. Therefore, without loss of generality, it suffices to prove the theorem under the assumption that every variable in C has degree at most an integer constant $d > 0$.

We say that a gate or a literal, g, *contains a variable* v if there is a path from the literal $\overline{v}$ to g. Denote by $V(g)$ the set of variables contained in g (if g is a literal, then $g = \overline{v}$, and $V(g) = \{v\}$). Let v be a variable contained in a gate g. We call v a *zero-variable* for g if assigning v the value 1 falsifies g; otherwise, v is called a *nonzero-variable* for g. In particular, a zero-variable (resp. nonzero-variable) for the output gate of C is a zero-variable (resp. nonzero-variable) for C, as previously defined.

When $t = 2$, every OR-gate incoming to the output gate of C contains at least two literals. Keep only two literals for each such OR-gate and remove the edges from the other literals to the OR-gate (without removing the literals from C). The problem asking whether C has a satisfying assignment becomes equivalent to the INDEPENDENT SET problem on multigraphs of degree bounded by d, which can be easily seen to have a solution of size $\Omega(n)$. By assigning 1 to the variables in the independent set and 0 to the remaining variables, the circuit is satisfied. The statement follows.

For simplicity of the presentation and to avoid repetition, the proof of the other base case when $t = 3$ (we induct on $t - 2$) will be combined with the proof of the inductive step, with the

[1]The constant d depends on the hidden constant in the upper bound $O(n)$ on the number of occurrences.

understanding that when $t = 3$ the inductive hypothesis does not apply, as explained later in the proof. Assume in what follows that $t \geq 3$, and that the statement is true for any circuit of depth smaller than t that satisfies the statement of the theorem.

First, observe that in the case when $d = 1$, C has a satisfying assignment in which at least $n/2$ variables are assigned 1. This can be seen as follows. Let $g_1, g_2, \ldots, g_r$ be the vertices incoming to the output gate of C. Since C has no zero-variables, each g_i, for $1 \leq i \leq r$, must be an OR-gate having at least two vertices incoming to it; we use a vertex here to denote a gate or a literal. From each g_i, pick a vertex v_i incoming to it that contains at most half of the variables contained in g_i; this can be done since every literal in C occurs exactly once. By assigning all variables in v_i, for $i = 1, \ldots, r$, the value 0, and all the remaining variables in C the value 1, we obtain an assignment that satisfies C, and in which at least half of the variables are assigned 1.

Suppose now that $d \geq 2$. Consider the following procedure:

Fix a variable in C; without loss of generality, let it be x_1 and let $g_1, g_2, \ldots, g_l$, where $l \leq d$, be the OR-gates incoming to the output AND-gate of C that contain x_1. For an arbitrary g_i, $1 \leq i \leq l$, if assigning x_1 the value 1 falsifies g_i then x_1 would be a zero-variable for the circuit, which is not possible. Therefore, there must exist an AND-gate or a literal, denoted w_i^1, incoming to g_i that is not falsified by assigning x_1 the value 1. Let U be the set of variables consisting of x_1 plus all the variables contained in $w_1^1, w_2^1, \ldots, w_l^1$. Consider the following cases:

Case 1. If $|U| \leq nd/(d+1)$, then assign x_1 the value 1, and the other variables in U the value 0. Every w_i^1, and hence every g_i, for $i = 1, \ldots, l$, is satisfied by this assignment. Afterwards, every g_i can be removed, and the resulting circuit has at least $n - nd/(d+1) = n/(d+1)$ variables left.
Case 2. If $|U| > nd/(d+1)$, then one of $w_1^1, w_2^1, \ldots, w_l^1$ contains at least $\frac{nd/(d+1)}{l} \geq \frac{nd/(d+1)}{d} = n/(d+1)$ variables in U; without loss of generality, let w_1^1 be such an AND-gate/literal. We further distinguish the following subcases:

2.1. If at most half of the variables of w_1^1 are zero-variables of w_1^1 (note that this case does not apply when $t = 3$, because when $t = 3$ all variables contained in w_1^1 are zero-variables of w_1^1), then assign the zero-variables of w_1^1 the value 0. Afterwards, w_1^1 is a $(t-2)$-level circuit of at least $n/(2d+2)$ nonzero-variables. Applying the inductive hypothesis to w_1^1, we know that w_1^1 has a satisfying assignment with at least $\log^{(d^{t-2})}(\frac{n}{2d+2})$ variables assigned 1. This means that in the antimonotone circuit, if we assign 0 to all but these variables, w_1^1 is satisfied and so is g_1, which can then be removed. Now the resulting circuit C has at least $\log^{(d^{t-2})}(\frac{n}{2d+2})$ variables left, whose degree is at most $d - 1$ because they are all incoming to gates in T_{g_1}, which is removed.

2.2. If any of the AND-gates or literals incoming to g_1, say w_1^2, shares fewer than $n/(2d + 2)$ variables with w_1^1, then $|V(w_1^1) \setminus V(w_1^2)| \geq n/(2d + 2)$. Assigning 0 to all variables in $V(w_1^2)$ will satisfy w_1^2 and hence will satisfy g_1, which can then removed. So the circuit C will have at least $n/(2d + 2)$ variables (in $V(w_1^1) \setminus V(w_1^2)$), whose degree is at most $d - 1$ (because g_1 is satisfied and removed).

2.3. Now assume that each AND-gate incoming to g_1 shares at least $n/(2d+2)$ variables with w_1^1, and hence each contains at least $n/(2d+2)$ variables. Since the total number of occurrences of the literals in C is at most dn, there are at most $\frac{dn}{n/(2d+2)} = 2d(d+1)$ AND-gates incoming to g_1. Let γ be the number of variables such that each is a nonzero-variable for at least one AND-gate incoming to g_1. We distinguish two subcases:

2.3.1. If $\gamma \geq n/(2d+2)$, then there exists an AND-gate incoming to g_1, denoted by w', that contains at least $\frac{n/(2d+2)}{2d(d+1)} = \frac{n}{4d(d+1)^2}$ nonzero-variables. (Note that this case does not apply when $t = 3$, when every variable is a zero-variable for every AND-gate incoming to g_1 that the variable is contained in.) By a similar argument to that made in 2.1, we apply the inductive hypothesis to w'. Afterwards, the circuit C has at least $\log^{(d^{t-2})}(\frac{n}{4d(d+1)^2})$ variables, whose degree is at most $d-1$.

2.3.2. If $\gamma < n/(2d+2)$, assign 0 to every nonzero-variable contained in a gate that is incoming to g_1. The remaining variables of g_1 are zero-variables of the AND-gates (or literals) incoming to g_1 (by our simplification rules mentioned in Section 2). In other words, what results of g_1 is an OR-gate of the form: $w_1^1 \vee w_1^2 \vee \ldots \vee w_1^s$, where $s \leq 2d(d+1)$ and each w_1^j is a literal or an AND-gate whose incoming edges are all from literals. Note that there are at most $2d(d+1)$ AND-gates (or literals) in g_1 and w_1^1 has at least $n/(2d+2)$ variables left. Denote by U_j the set of variables shared by all $w_1^1, \ldots, w_1^j$:

$$U_j = V(w_1^1) \cap \ldots \cap V(w_1^j).$$

Consider the following process:
If $|U_2| \leq |U_1|/2$, then $|U_1 \setminus U_2| \geq |U_1|/2 = |V(w_1^1)|/2 \geq n/(4d+4)$. Assign 0 to all variables except those in $U_1 \setminus U_2$, we have a circuit of at least $n/(4d+4)$ variables, whose degree is at most $d-1$ because g_1 is satisfied and removed. If $|U_2| \geq |U_1|/2$, then proceed similarly: if $|U_3| \leq |U_2|/2$, then assign 0 to all variables except those in $U_2 \setminus U_3$, we have a circuit of at least $|U_2|/2$ variables, whose degree is at most $d-1$ because g_1 is satisfied and removed. Proceed in this fashion, so we either have a circuit of at least $\frac{n/(d+1)}{2^s} \geq \frac{n/(d+1)}{2^{2d(d+1)}}$ variables whose degree is at most $d-1$, or we end up with $|U_s| > \frac{n/(d+1)}{2^s} \geq \frac{n/(d+1)}{2^{2d(d+1)}} > 0$, which is impossible because any variable in U_s is a zero-variable of C.

This completes the description of the procedure.

Note that no zero-variables are created in any of the above cases. This is true because in all cases except **Case 1**, we assign the variables in C only the value 0, which does not create zero-variables, while in **Case 1**, x_1 is assigned 1, but every gate containing x_1 is removed (except the output gate). Note also that the second base case of $t = 3$ can be treated by the above process because $t = 3$ is only possible in **Case/Subcase** 1, 2.2, and 2.3.2, all of which do not rely on the inductive hypothesis.

So in one iteration of the above process, we either: (1) reduce the number of variables from n to $n/(d+1)$ and assign 1 to a variable (**Case-1** operation), or (2) reduce the number of variables from n to a number of variables that is at least $\min\{\log^{(d^{t-2})}(\frac{n}{2d+2}), n/(2d+2), \log^{(d^{t-2})}(\frac{n}{4d(d+1)^2}), \frac{n/(d+1)}{2^{2d(d+1)}}\} = \log^{(d^{t-2})}(\frac{n}{4d(d+1)^2})$, and reduce the degree of the variables by 1 (**Case-2** operation). Afterwards, we can repeat the process until $f(n)$ variables are assigned 1, or until the degree of the variables in the circuit is at most 1. After a number of iterations, if $f(n)$ variables are already assigned 1 and the circuit is not empty, then we can assign 0 to all other variables and we are done. If the degree of the variables in the circuit is at most 1, then as we mentioned at the beginning of the proof, at least half of the remaining variables can be assigned 1. So it remains to be shown that when the degree of the variables in the circuit is at most 1, there are at least $2f(n)$ variables left.

In any sequence of iterations, **Case-1** operation is applied at most $f(n)$ times and **Case-2** operation is applied at most d times. Let $g(n) = n/(d+1)$ and $h(n) = \log^{(d^{t-2})}\left(\frac{n}{4d(d+1)^2}\right)$. Note that $g(h(n)) \leq h(g(n))$, *i.e.*, $g \circ h \leq h \circ g$. So the number of variables in the circuit after any sequence of iterations is at least:

$$\underbrace{g \circ \ldots \circ g}_{f(n)} \circ \underbrace{h \circ \ldots \circ h}_{d}(n).$$

Note that $h(n) = \log^{(d^{t-2})}\left(\frac{n}{4d(d+1)^2}\right) \geq \log^{(d^{t-2})}\log n = \log^{(d^{t-2}+1)} n$. So we have:

$$\underbrace{h \circ \ldots \circ h}_{d}(n) \geq \underbrace{\log^{(d^{t-2}+1)} \circ \ldots \circ \log^{(d^{t-2}+1)}}_{d} n = \log^{(d(d^{t-2}+1))} n = \log^{(d^{t-1}+d)} n. \tag{1}$$

On the other hand:

$$\underbrace{g \circ \ldots \circ g}_{f(n)}(n) = n/(d+1)^{f(n)} = n/(d+1)^{\log^{(d^t)} n} \geq n/\log^{(d^t-2)} n > \log n. \tag{2}$$

Finally, since $d \geq 2$ and $t \geq 3$, we have:

$$\underbrace{g \circ \ldots \circ g}_{f(n)} \circ \underbrace{h \circ \ldots \circ h}_{d}(n) \geq \log(\log^{(d^{t-1}+d)} n) = \log^{(d^{t-1}+d+1)} n \geq 2\log^{(d^t)} n = 2f(n). \tag{3}$$

This means that in any sequence of iterations, we either assign 1 to $f(n)$ variables or end up with at least $2f(n)$ variables of degree at most 1, in which case the circuit can be satisfied by assigning 1 to $f(n)$ variables. So in either case, the statement is true for circuits of depth $t \geq 2$.

This completes the proof. □

Lemma 3.5. *Let $c > 0$ be a constant. The* WSAT$^-[t]$ *($t \geq 2$) problem on circuits of genus $g(n) = \Omega(n^c)$, where n is the number of variables in the circuit, is* W$[t]$*-complete for odd t and* W$[t-1]$*-complete for even t.*

Proof. To prove the hardness result in the theorem, we show that WSAT$^-[t]$ is FPT-reducible to WSAT$^-[t]$ on circuits of genus $g(n) = \Omega(n^c)$. Since WSAT$^-[t]$ is W$[t]$-hard for odd t, and W$[t-1]$-hard for even t, the hardness result follows. Suppose that $g(n) = c'n^c$, for some constant $c' > 0$.

Let (C_0, k) be an instance of WSAT$^-[t]$, where C_0 is a Π_t^- circuit and k is the parameter. Suppose that C_0 has n_0 variables and m_0 gates (including the variables). Therefore, the genus of C_0 is at most m_0^2. If $m_0^2 \leq c'n_0^c$, then the FPT-reduction outputs the instance (C, k), where $C = C_0$. If $m_0^2 > c'n_0^c$, let C be the circuit obtained from C_0 by adding $\lceil (m_0^2/c')^{(1/c)} \rceil - n_0$ new negative literals that are incoming to the output AND-gate of C_0. The FPT-reduction outputs the instance (C, k). Obviously, the genus of C is at most that of C_0, which is at most m_0^2. It can be easily verified that the genus of C, in both cases, is at most $c'n^c$, where n is the number of variables in C. Noting that the new literals (if added) must be assigned value 1, and hence their corresponding variables value 0, by any satisfying assignment of C, we conclude that C_0 has a weight-k satisfying assignment if and only if C has a weight-k satisfying assignment. It follows that the above reduction is an FPT-reduction from WSAT$^-[t]$ to WSAT$^-[t]$ on circuits of genus $g(n) = \Omega(n^c)$.

The completeness of the problem follows from the membership of WSAT$^-[t]$ in W$[t]$ for odd t, and in W$[t-1]$ for even t. □

Now we have a tight result on the parameterized complexity of the WSAT$^-[t]$ problem with respect to the genus of the circuit.

Theorem 3.6. *The* WSAT$^-[t]$ *($t \geq 2$) problem on circuits of genus $g(n) = n^{o(1)}$ (n is the number of variables) is* FPT*, and is* W$[t]$*-complete for odd t and* W$[t-1]$*-complete for even t if $g(n) = n^{\Omega(1)}$.*

Proof. Let $g(n) = n^{o(1)} = n^{1/\mu(n)}$, where $\mu(n)$ is a complexity function, and let (C, k) be an instance of the WSAT$^-[t]$ ($t \geq 2$) problem on circuits of genus $g(n)$. By Proposition 3.2, we can assume that C has no zero-variables, and that the number of variables n in C is least $g(n)$. By Proposition 3.3, we may assume that the number of occurrences of the literals in C is $O(n)$; if this is not the case then the genus of the circuit is not upper bounded by $g(n)$, and we reject the instance. By Theorem 3.4, C has a satisfying assignment in which at least $f(n)$ variables are assigned the value 1, where $f(n)$ is the function given in the lemma. Therefore, if $k \leq f(n)$ then we accept the instance (C, k); otherwise, $k > f(n)$ and in FPT-time we can decide the instance by a brute-force algorithm that enumerates every weight-k assignment. The hardness result follows from Lemma 3.5. □

4 The monotone case

In this section we investigate the parameterized complexity of the WSAT$^+[t]$ problem, where $t \geq 2$ is an integer, with respect to the genus of the circuit.

Proposition 4.1. *Let (C, k) be an instance of* WSAT$^+[t]$ *($t \geq 2$) such that C has genus $g(n) = n^{o(1)}$. There is an* FPT*-time algorithm that reduces (C, k) to $h(k)n^{O(1)}$ many instances (C', k') of* WSAT$^+[t]$*, where h is a complexity function and $k' \leq k$, such that (C, k) is a yes-instance if and only if at least one of the instances (C', k') is, and such that each instance (C', k') satisfies that: (1) the number of critical gates in C' is at most $2k'$, (2) every variable in C' is incoming to gates in at most two subtrees T_p, T_q of T'_C rooted at critical gates p, q in C', and (3) the genus of C' is at most $g(n)$.*

Proof. Let $g(n)$ be a complexity function such that $g(n) = n^{o(1)}$. Since $g(n) = n^{o(1)}$, $g(n) \leq n^{1/\mu(n)}$ for some complexity function $\mu(n)$.

Let (C, k) be an instance of WSAT$^+[t]$, where C is a Π_t^+ circuit with set of variables $X = \{x_1, \ldots, x_n\}$, and k is the parameter. If more than k variables are incoming to the output gate of C, then clearly C has no satisfying assignment of weight k, and we reject the instance (C, k). Otherwise, we can assign the value 1 to the variables incoming to the output-gate of C, remove these variables, and update C and k accordingly. So we may assume, without loss of generality, that C has no variables incoming to its output gates, and that all gates incoming to the output gates are OR-gates (by the simplification rules discussed in Section 2), and hence are critical gates.

For each critical gate p in C, consider the subtree T_p of T_C. In the case when $t = 2$, this subtree is trivial, and consists of gate p. We form an auxiliary graph $\mathcal{B}$ as follows. Starting at each critical gate p, we contract the edges in T_p to form a single vertex p' whose incoming variables are the variables that are incoming to at least one gate in T_p. Note that if a variable is incoming to several gates in T_p, then there will be multiple edges between p' and this variable. Let $\mathcal{G}$ be the set of vertices resulting from contracting each tree T_p corresponding to a critical gate p in C. Let $\mathcal{B} = (\mathcal{G}, X)$ be the underlying undirected bipartite graph resulting from this contraction with the multiple edges removed. That is, there is an (undirected) edge in $\mathcal{B}$ between a variable $x_i \in X$ and a gate p' in $\mathcal{G}$ if and only if x_i is incoming to some gate in T_p. Clearly, the genus of $\mathcal{B}$ is at most $g(n)$. Observe that since each critical gate p must be satisfied by every assignment that satisfies C, for any vertex p' in $\mathcal{G}$, at least one variable incident to p' in $\mathcal{B}$ must be assigned 1 in any truth assignment satisfying C. Let $n_g = |\mathcal{G}|$.

We partition the variables in X into two sets: $X_{\geq 3}$ that consists of each variable in X whose degree in $\mathcal{B}$ is at least 3, and $X_{\leq 2}$ consisting of each variable in X whose degree in $\mathcal{B}$ is at most 2. Let $n_3 = |X_{\geq 3}|$ and $n_2 = |X_{\leq 2}|$. By defining a multihypergraph whose vertex-set is $\mathcal{G}$, and whose hyperedges correspond to the neighborhoods of the variables in $X_{\geq 3}$, we obtain from Lemma 2.1 that $n_3 \leq 2n_g + 4g(n)$; if the preceding upper bound on n_3 does not hold, then we reject the instance (this means that the genus of the circuit is not at most $g(n)$). We perform the following search-tree algorithm $\mathcal{A}$ that distinguishes two cases:

Case 1. $n_g \leq n^{1/\mu(n)}$. In this case we have $n_3 \leq 2n_g + 4g(n) \leq 6n^{1/\mu(n)}$. The number of subsets of $X_{\geq 3}$ of size at most k is at most $\Sigma_{i=0}^{k}\binom{n_3}{i} \leq kn_3^k \leq k \cdot (6n^{1/\mu(n)})^k$. We try each such subset of $X_{\geq 3}$ as a candidate subset of variables that will be assigned value 1 by a satisfying assignment of weight k. For each such candidate subset S, we update the gates in C in a natural way according to the partial assignment assigning the variables in S the value 1, and those in $X_{\geq 3} \setminus S$ the value 0. We remove all variables in $X_{\geq 3}$ from C, and update C and k appropriately. Since each remaining variable is in $X_{\leq 2}$, each variable can satisfy at most 2 critical gates, and hence if the number of critical gates in C is more than $2k$, then we can reject the resulting instance (C, k). Therefore, for each instance resulting from the enumeration of such a subset S of $X_{\geq 3}$, either the number of remaining critical gates in C is more than $2k$ and we reject the instance since k variables in $X_{\leq 2}$ cannot satisfy all the critical gates of C, or the number of critical gates in C is at most $2k$. Since the number of candidate subsets of $X_{\geq 3}$ is at most $k \cdot (6n^{1/\mu(n)})^k$, which can be enumerated in FPT-time by Lemma 2.4, the statement of the theorem follows.

Case 2. $n_g > n^{1/\mu(n)}$. Let G be the subgraph of $\mathcal{B}$ induced by the set of vertices in $\mathcal{G}$ plus those in $X_{\geq 3}$. Since $n_3 \leq 2n_g + 4g(n) \leq 6n_g$, the number of vertices in G is at most $7n_g$. Since the genus of G is at most $g(n)$, by Lemma 2.2, the number of edges in G is at most $21n_g + 6g(n) \leq 27n_g$. Let $Y_{\geq 3}$ be the set of variables in $X_{\geq 3}$ of degree at least $27n_g/\log n$ in G. Since the number of edges in G is at most $27n_g$, it follows that $|Y_{\geq 3}| \leq \log n$. In time $(\log n)^k$, which is FPT-time by Lemma 2.4, we can enumerate each subset of $Y_{\geq 3}$ of size at most k as a candidate subset of variables that are assigned value 1 by a satisfying assignment of weight k. For each such *nonempty* candidate subset, C is updated appropriately (as in **Case 1** above) and k is decreased by at least the size of the subset, which is nonzero, and we can repeat the execution of the whole algorithm $\mathcal{A}$; this algorithm will be repeated at most k times. If the candidate subset is empty, then along this branch we reject the instance (C, k) since C cannot be satisfied by an assignment of weight k. The preceding statement can be justified as follows. In any satisfying assignment, the critical gates, whose number is $n_g > n^{1/\mu(n)}$, must be satisfied. Since the chosen subset of $Y_{\geq 3}$ is empty, we are working under the assumption that no variable in $Y_{\geq 3}$ is assigned 1 by any satisfying assignment. Therefore, the variables assigned 1 by any satisfying assignment must be chosen from $X_{\geq 3} - Y_{\geq 3}$ or from $X_{\leq 2}$. Each variable in $X_{\geq 3} - Y_{\geq 3}$ can satisfy at most $27n_g/\log n$ critical gates in C, and each variable in $X_{\leq 2}$ can satisfy at most 2 critical gates. Therefore, k variables from $(X_{\geq 3} - Y_{\geq 3}) \cup X_{\leq 2}$ can satisfy at most $27kn_g/\log n < n_g$ critical gates in C, and hence cannot satisfy C. We assumed here that $k < \log n/27$; otherwise, we can decide the instance in FPT-time.

It follows that the algorithm $\mathcal{A}$ outlined above runs in FPT-time, and either solves the instance (C, k), or reduces it to $h(k)n^{O(1)}$ many instances (C', k') $(k' < k)$, such that (C, k) is a yes-instance if and only if at least one of the instances (C', k') is, and such that each of the instances (C', k') satisfies conditions (1), (2), and (3) in the statement of the theorem. $\square$

The proof of the following lemma is similar to that of Lemma 3.5:

Lemma 4.2. *Let $c > 0$ be a constant. The* WSAT$^+[t]$ *($t \geq 2$) problem on circuits of genus $g(n) = \Omega(n^c)$ is* W$[t]$*-complete for even t and* W$[t-1]$*-complete for odd t.*

Theorem 4.3. *The* WSAT$^+$ *problem on circuits of genus $g(n)$ is* FPT *if $g(n) = n^{o(1)}$, and is* W$[2]$*-complete if $g(n) = n^{\Omega(1)}$.*

Proof. By Proposition 4.1, in FPT-time we can reduce an instance (C, k) of WSAT$^+$ on circuits of genus $g(n) = n^{o(1)}$ to $h(k)n^{O(1)}$ many instances (C', k') of WSAT$^+$, such that each instance (C', k') satisfies the properties described in Proposition 4.1. It suffices to show that we can decide each such instance (C', k') in FPT-time. First, observe that since each subtree T_p rooted at a critical gate p consists of a single critical gate of C', each variable in C' has outdegree at most 2; that is, each variable in C' is incoming to at most two gates in C'. For two variables x_i and x_j in C', if the set of gates that x_i is incoming to is a subset of that of x_j, then we say that x_j *dominates* x_i. We perform the following reductions. If more than k' variables are incoming to the output gate of C', then C' has no satisfying assignment of weight k', and we reject (C', k'). Otherwise, we assign the value 1 to the variables incoming to the output gate of C', remove them, and update C' and k' accordingly. For any two 2-literal gates that have the same pair of variables incoming to them, we remove one of the two gates from C'. So assume, without loss of generality, that in the instance (C', k') the circuit C' contains no variables incoming to its output gate, and that there are no two 2-literal gates in C' with the same pair of variables incoming to them. For every two variables x_i and x_j in C, if x_i dominates x_j then remove x_j. After applying the previous reductions, it is easy to see that the number of degree-1 variables is at most $2k'$, and the number of degree-2 variables is at most $\binom{2k'}{2}$. Therefore, the resulting circuit has size $O(k'^2)$, and in FPT-time we can decide (C', k'). The hardness result follows from Lemma 4.2. □

The rest of this section handles the cases when $t > 2$. We first have the following definition.

Definition 4.4. *Let $G = (V, E)$ be a graph. A tree decomposition of G is a pair $(\mathcal{V}, \mathcal{T})$ where $\mathcal{V}$ is a collection of subsets of V such that $\bigcup_{X_i \in \mathcal{V}} = V$, and $\mathcal{T}$ is a tree whose node set is $\mathcal{V}$, such that:*

1. *for every edge $\{u, v\} \in E$, there is an $X_i \in \mathcal{V}$, such that $\{u, v\} \subseteq X_i$;*
2. *for all $X_i, X_j, X_k \in \mathcal{V}$, if the node X_j lies on the path between the nodes X_i and X_k in the tree $\mathcal{T}$, then $X_i \cap X_k \subseteq X_j$;*

   The width of the tree decomposition $(\mathcal{V}, \mathcal{T})$ is defined to be $\max\{|X_i| \mid X_i \in \mathcal{V}\} - 1$. The treewidth of the graph G is the minimum tree width over all tree decompositions of G.

Consider an $(r \times r)$-grid. A *corner vertex* of the grid is a vertex of the grid of degree 2. By Γ_r (see [14]) we denote the graph obtained from the $(r \times r)$-grid as follows: construct first the graph Γ'_r by triangulating all internal faces of the $(r \times r)$-grid such that all internal vertices of the grid are of degree 6, and all non-corner external vertices of the grid are of degree 4. Notice that Γ'_r is unique up to isomorphism. Also, two of the corners of the initial grid have degree 2 in Γ'_r; let x be one of them. Γ_r is obtained from Γ'_r by adding all the edges having x as an endpoint and a vertex of the external face of the grid that is not already a neighbor of x as the other endpoint (see Figure 2 for an illustration of Γ_6). Observe again that Γ_r is unique up to isomorphism. The following lemma is implied from Lemma 6 in [14]:

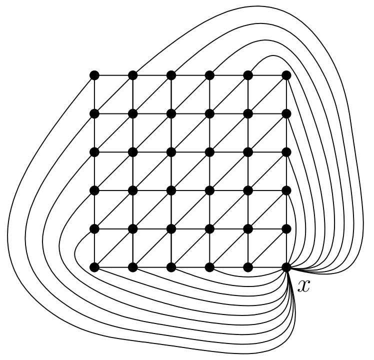

Figure 2: The graph Γ_6.

Lemma 4.5 (Lemma 6 in [14]). *Let G be a graph of genus g, and let r be any positive integer. If G excludes Γ_r as an s-contraction, then the treewidth of G is at most $(2r+4) \cdot (g+1)^{3/2}$.*

We note that Lemma 6 in [14] uses the notion of *smooth contraction* instead of s-contraction, but it is certainly true that if G excludes Γ_r as an s-contraction then it excludes Γ_r as a smooth contraction as well.

Lemma 4.6. *Let (C,k) be an instance of* WSAT$^+[t]$ *$(t \geq 2)$ such that C has genus $g(n)$ and at most $2k$ critical gates. Let C^- be the circuit resulting from C after removing the output gate. The treewidth of the underlying graph of C^- is $O(\sqrt{k} \cdot g^{3/2}(n))$.*

Proof. We show that there exists an integer constant $c > 0$ such that the underlying graph of C^- excludes $\Gamma_{(c \cdot \lceil \sqrt{k} \rceil)}$ as an s-contraction. The result will then follow from Lemma 4.5.

Suppose, to get a contradiction, that for every integer constant $c > 0$ the underlying graph of C^-, G, contains $\Gamma_{(c \cdot \lceil \sqrt{k} \rceil)}$ as an s-contraction. Since the depth of C is at most t, every literal and gate in C^- is within distance (*i.e.*, length of a shortest path) at most t from some critical gate of C. Let S be the set of vertices in G, each of which either corresponds to a critical gate of C or to an s-contraction of a critical gate of C, and note that $|S| \leq 2k$. Clearly, every vertex in G must be within distance at most t from one of the vertices in S. Since t is a constant, it is easy to see that the number of vertices in G within distance t from at least one vertex in S is at most $c'k$, for some integer constant $c' > 0$ that is independent of c and k. On the other hand, the number of vertices in G is $(c \cdot \lceil \sqrt{k} \rceil)^2 \geq c^2 k$. By choosing c large enough (larger than $\sqrt{c'}$), we arrive at a contradiction since there would be vertices in G that are not within distance t from any vertex in S (note that k is assumed to be larger than any pre-specified constant). □

Using the above lemma, we can design a dynamic programming approach based on tree decomposition. To simplify the dynamic programming algorithm, we use the notion of a *nice tree decomposition* defined as follows.

A (rooted) tree decomposition $(\mathcal{V}, \mathcal{T})$ is *nice* if it satisfies the following conditions:

1. Each node in the tree $\mathcal{T}$ is either a leaf (has no children) or has at most two children.

2. If a node X_i has two children X_j and X_k in the tree $\mathcal{T}$, then $X_i = X_j = X_k$; in this case node X_i is called a *join node.*

3. If a node X_i has only one child X_j in the tree $\mathcal{T}$, then either $|X_i| = |X_j| + 1$ and $X_j \subset X_i$, and in this case X_i is called an *insert node*; or $|X_i| = |X_j| - 1$ and $X_i \subset X_j$, and in this case X_i is called a *forget node.*

Proposition 4.7. *Let C be a Π_t^+ circuit, and let $G = (V, E)$ be the undirected underlying graph of C with the output gate removed. If a tree decomposition for G of N nodes and treewidth ω is given, then a minimum weight satisfying assignment of C can be computed in time $O(9^\omega N^{O(1)})$.*

Proof. Let $\mathcal{X} = \langle\{X_i \mid i \in \mathcal{T}\}, \mathcal{T}\rangle$ be a *nice* tree decomposition for the graph G. We assume that the tree decomposition is nice; otherwise, based on $\mathcal{T}$ we can compute a nice tree decomposition of the same width in polynomial time in the size of $\mathcal{T}$ [20]. To simplify the notation, we call a vertex in G a "variable" (resp. a "gate") if its corresponding vertex in C is a variable (resp. a gate).

We use a dynamic programming approach to compute a minimum weight satisfying assignment for C. Let $X_i = (x_{i_1}, \dots, x_{i_{n_i}})$ be a bag in $\mathcal{X}$, where each of $x_{i_1}, \dots, x_{i_{n_i}}$ is either a variable or a gate. For an $x_{i_r} \in X_i$, if x_{i_r} is a variable we assign it either the color "white", meaning that its value is 0/FALSE, or the color "black", meaning that its value is 1/TRUE; if x_{i_r} is a gate, we assign it one of three colors: "black", "gray", and "white". Here are the interpretations of the colors, and the rules for assigning them to the gates:

- black: TRUE and justified. For an OR-gate, this means that one of the vertices incoming to x_{i_r} is colored black or gray; for an AND-gate, this means that all the vertices incoming to x_{i_r} are black or gray.
- gray: TRUE but unjustified. For an OR-gate, this means that every vertex incoming to x_{i_r} is either uncolored or is colored white; for an AND-gate, this means that at least one vertex incoming to x_{i_r} is uncolored, and the colored vertices incoming to x_{i_r} are black or gray.
- white: FALSE, either justified or unjustified. For an OR-gate, this means that every vertex incoming to x_{i_r} is either uncolored or is colored white; for an AND-gate, this means that one of the vertices incoming to x_{i_r} is either uncolored or is white.

A vector $c_i = (c_{i_1}, \dots, c_{i_{n_i}})$ is called a *coloring* of the bag X_i, where c_{i_r} is the color of x_{i_r}. The weight of a coloring c_i of a bag X_i, denoted $W(c_i)$, is the minimum number of variables assigned TRUE in the graph induced by the subtree of $\mathcal{T}$ rooted at X_i, under the restriction that c_i is the coloring of X_i.

The dynamic programming algorithm will compute valid colorings of the bags in $\mathcal{T}$ and their weights in a bottom-up fashion starting at the leaves of $\mathcal{T}$. During this process, we check for validity of the colorings according to the rules of assigning the colors, and purge invalid ones. Additionally, if a gate in G is critical and is colored white, then the coloring is also invalid and purged.

First, for each leaf bag in the tree decomposition, we compute the valid colorings and their weights for this bag. The valid colorings can be computed by enumerating all colorings and checking for their validity according to coloring rules. Next, we move up the tree from the leaves to the root, computing the colorings and their weights of a parent depending on the colorings and weights of its child (or children). We set the following ground rule regarding the coloring of a vertex shared by a parent (bag) and its child (bag):

> **Ground Rule**: If the shared vertex is a variable, then its color must be the same in the parent and in the child; if the shared vertex is a gate, then either its color is the same in the parent and in the child, or its color is gray (TRUE but unjustified) in the child and black (TRUE and justified) in the parent.

The ground rule is based on the following reasoning: A vertex that is colored black or white does not change its color in a valid coloring; an AND-gate colored gray can be *upgraded* (later) to black when all vertices incoming to it are colored black or gray; and an OR-gate colored gray can be *upgraded* to black when a vertex incoming to it becomes black or gray.

We distinguish three cases according to the types of the nodes in the tree decomposition.

0. Leaf node: Let X_i be the bag of a leaf node. We enumerate all colorings of the vertices in X_i. For each possible coloring c, we check its validity according to the coloring rules, and store it if it is valid.

1. Forget node: Let X_i be the bag of a forget node and $X_j = X_i \cup \{x\}$ be the bag of its child, where x is the vertex to be "forgotten". The colorings of X_i are the projection of the colorings of X_j. The weight of a coloring c of X_i is the minimum weight of the colorings of X_j that produce c. Note that by the time a gate g is to be forgotten, it will not be colored gray because by then all vertices incoming to g have been considered, and hence its color should not remain unjustified.

2. Insert node: Let X_i be the bag of an insert node and $X_j = X_i \setminus \{x\}$ be the bag of its child, where x is the vertex to be "inserted". We will extend the colorings of X_j by assigning x its possible color options. After inserting the new vertex and assigning it a color, a coloring may become invalid, and in which case the coloring is discarded. After inserting the new vertex and assigning it a color, it is possible that some gray gate may be upgraded to black, then it is updated as such. Note that upgrading the color of a vertex v from gray to black does not affect the colors of the vertex that v incoming to (by the coloring rules). The weight of a coloring c of X_i is the minimum weight of the colorings of X_j that produce c, plus one if the new vertex x is a variable and is assigned TRUE.

3. Join node: Let X_i be the bag of a join node and X_j, X_k be the bags of its children, where $X_i = X_j = X_k$. Let x be a vertex in X_i. If x is a variable, then the color of x must be the same in X_i, X_j, and X_k according to the **Ground Rule**. If x is a gate, the color of x can be the same in X_i, X_j, and X_k, or, according to the **Ground Rule**, one of the following cases applies: (1) x is black in X_i and X_j, and gray in X_k (or symmetrically, black in X_i and X_k, and gray in X_j), or (2) x is black in X_i, and gray in both X_j and X_k. In the following, we discuss these cases based on the type of the gate x.

   If x is an AND-gate, case (1) happens when all the vertices incoming to x are TRUE (either justified or unjustified) and all of them are in the subtree rooted at X_j (and hence X_i), but not all of them are in the subtree rooted at X_k. Case (2) happens when all the vertices incoming to x are TRUE (either justified or unjustified), and all of them are in the subtree rooted at X_i, but not all of them are in the subtree rooted at X_j or X_k.

   If x is an OR-gate, case (1) happens when a vertex incoming to x is TRUE in the resolved portion of the subtree rooted at X_j (and hence X_i), but it is not in the subtree rooted at X_k. Case (2) is impossible because if x has a vertex incoming to it that is colored black or gray, this vertex should appear in X_j or in X_k.

   In each of these cases, if a coloring c_i of X_i is produced from a coloring c_j of X_j and a coloring c_k of X_k, then $W(c_i) = \min(W(c_j) + W(c_k) - \#_1(c_i))$ over all colorings c_j and c_k that produce c_i, where $\#_1(c_i)$ is the number of variables assigned TRUE in coloring c_i.

Since each vertex can be colored with at most three colors, it is easy to see that Cases 0–2 above can be executed in $O(3^\omega N^{O(1)})$ time, and that Case 3 can be executed in $O(9^\omega N^{O(1)})$

time. Finally, the minimum weight satisfying assignment is the minimum weight of the colorings of the root. The total running time of the dynamic programming algorithm outlined above is $O(9^{\omega} N^{O(1)})$. It may be possible to obtain improvement on the running time of the dynamic programming algorithm above using the generalized subset convolution technique (see [27]). □

Theorem 4.8. *The* WSAT$^+[t]$ *problem* ($t > 2$) *on circuits of genus* $g(n) = o(\log^{2/3}(n))$ *is* FPT, *and is* W$[t]$*-complete for even* t *and* W$[t-1]$*-complete for odd* t *if* $g(n) = n^{\Omega(1)}$.

Proof. Let (C, k) be an instance of WSAT$^+[t]$ on circuits of genus $g(n) = \lambda(n)$, where $\lambda(n) = o(\log^{2/3}(n))$. By Proposition 4.1, in FPT-time we can reduce the instance (C, k) to $h(k)n^{O(1)}$ many instances (C', k') of WSAT$^+[t]$, where h is a complexity function of k and $k' \leq k$, such that (C, k) is a yes-instance if and only if at least one of the instances (C', k') is, and such that C' has genus at most that of C and C' has at most $2k'$ critical gates. Therefore, without loss of generality, we may assume that C has at most $2k$ critical gates. By Lemma 4.6, the treewidth of C is at most $c_1\sqrt{k} \cdot \lambda'(n)$, where $\lambda'(n) = o(\log n)$. Using the algorithm in [3], we can decide if the treewidth of C is at most $c_2\sqrt{k} \cdot \lambda'(n)$ for some fixed constant $c_2 > 0$ (if not, the genus does not satisfy the given upper bound and we reject the instance), and if so, the algorithm in [3] returns a tree decomposition of C of width $c_4\sqrt{k} \cdot \lambda'(n)$, for some constant $c_4 > 0$, in time $2^{O(\sqrt{k} \cdot \lambda'(n))}|C|^{O(1)} = 2^{\sqrt{k} \cdot o(\log n)}|C|^{O(1)} = n^{o(1) \cdot \sqrt{k}}|C|^{O(1)}$, which is FPT-time by Lemma 2.4. By Proposition 4.7, we can decide (C, k) in time $2^{O(\sqrt{k} \cdot \lambda'(n))}|C|^{O(1)}$, which is FPT-time following the same preceding analysis. The hardness result follows from Lemma 4.2. □

The above approach can be extended to prove that WSAT$^+[3]$ on circuits of genus $n^{o(1)}$ is FPT, thus obtaining tight results for $t = 3$ as well. Because the approach is very similar, to avoid repetition, we only sketch the proof and emphasize the differences.

Theorem 4.9. *The* WSAT$^+[3]$ *problem on circuits of genus* $g(n)$ *is FPT if* $g(n) = n^{o(1)}$, *and* $W[2]$*-complete if* $g(n) = n^{\Omega(1)}$.

Proof. We will sketch the proof. First, using a similar proof to that of Lemma 4.6, we can show that a circuit whose genus is $n^{o(1)}$ and having at most $2k$ critical gates has treewidth $n^{o(1)}$. Using the result of Bodlaender et al. [5], we can compute a tree decomposition of the underling graph of the circuit of treewidth $O(n^{o(1)} \cdot \log n)$.

We then modify the dynamic programming approach in Proposition 4.7 to work on a tree decomposition of treewidth $O(n^{o(1)} \cdot \log n)$. The basic observation needed for this proof is that we can enumerate all necessary configurations of the vertices in the bags of size $O(n^{o(1)} \cdot \log n)$ in fpt-time. This can be seen as follows. For $t = 3$, the circuit consists of an output gate, at most $2k$ critical gates that must be satisfied, level-2 AND-gates, and variables. Therefore, each bag of the tree decomposition contains within it critical gates, level-2 AND-gates, and variables. Observe the following:

1. Critical gates must be satisfied, and hence no enumeration is needed to their status.

2. Since we are looking for a weight-k satisfying assignment, we only need to enumerate subsets of variables of size at most k in a bag (in the tree decomposition) whose size is $n^{o(1)} \cdot \log n$. Hence, we can enumerate all subsets of at most k variables in any bag in time $O(n^{o(1)k} \cdot (\log n)^k)$, which is FPT by Lemma 2.4.

3. A critical gate is satisfied if and only if at least one of its incoming level-2 AND-gates (or a variable) is satisfied. Therefore, we only need to enumerate the set of level-2 AND-gates

satisfying the condition that no two level-2 AND-gates in the set are incident to the same critical gate. Therefore, we only need to enumerate subsets of size at most k from the set of level-2 AND-gates in a bag, whose size is $n^{o(1)} \cdot \log n$.

It follows from Lemma 2.4 that we can enumerate all necessary configurations of the vertices in a bag in fpt-time. □

5 Applications

We show applications of the above complexity results to some natural problems. The parameterized RED-BLUE NONBLOCKER problem is: Given a bipartite graph with one partition colored red and the other blue and a parameter k, decide whether or not there exists a set S of k red vertices such that every blue vertex has a red neighbor not in S. The RED-BLUE DOMINATING SET problem is: Given a bipartite graph with one partition colored red and the other blue and a parameter k, decide whether or not there exists a set S of k red vertices such that every blue vertex has a red neighbor in S (*i.e.*, a set of red vertices of size k that dominates all blue vertices). The HITTING SET problem is: Given a hypergraph H and a parameter k, decide whether or not there exists a set C of k vertices that intersects every hyperedge in H (*i.e.*, the VERTEX COVER problem on hypergraphs). The INDEPENDENT SET ON HYPERGRAPHS problem is: Given a hypergraph H and a parameter k, decide whether or not there exists a set I of k vertices such that no hyperedge in H is a subset of I. The SET COVER problem is: Given a hypergraph H and a parameter k, decide if there are at most k hyperedges whose union is $V(H)$. Recall that, by definition, the genus of a hypergraph is the genus of its bipartite incidence graph (see Section 2).

Theorem 5.1. *The parameterized* RED-BLUE DOMINATING SET, HITTING SET, *and* SET COVER *are* FPT *on graphs/hypergraphs of genus $N^{o(1)}$ and* W[2]*-complete on graphs/hypergraphs of genus $N^{\Omega(1)}$, where N is the number of red vertices in* RED-BLUE DOMINATING SET*, the number of vertices in* HITTING SET*, and the number of hyperedges in* SET COVER.

Proof. The FPT results follow by simple and standard FPT-reductions that preserve the genus (and N) from the RED-BLUE DOMINATING SET, HITTING SET and SET COVER problems to WSAT$^+$ combined with Theorem 4.3. We briefly sketch these standard reductions, and refer the reader to [12, 13] for more details about these reductions.

The reduction from RED-BLUE DOMINATING SET on graphs of genus $N^{o(1)}$ to WSAT$^+$ corresponds to every red vertex in the bipartite graph a variable and to every blue vertex a critical (OR) gate; an edge is drawn between a variable and a gate if and only if an edge exists between the corresponding vertices in the graph. The reduction from HITTING SET on hypergraphs of genus $N^{o(1)}$ to WSAT$^+$ corresponds to every vertex in the hypergraph a variable and to every hyperedge a critical gate; an edge exists between a variable and a gate if and only if the vertex corresponding to the variable is contained in the hyperedge corresponding to the gate. The reduction from SET COVER on hypergraphs of genus $N^{o(1)}$ to WSAT$^+$ corresponds to every hyperedge in the hypergraph a variable and to every vertex in the hypergraph a critical gate; an edge exists between a variable and a gate if and only if the hyperedge corresponding to the variable contains the vertex corresponding to the gate. The parameter k remains unchanged in all the aforementioned reductions. It is easy to verify that the aforementioned reductions are FPT-reduction that preserve the genus of the the underlying graph/hypergraph.

The W[2]-hardness results follow by standard FPT-reductions that preserve the genus from WSAT$^+$ on circuits of genus $N^{\Omega(1)}$ to RED-BLUE DOMINATING SET, HITTING SET, and SET COVER combined with Theorem 4.3; these reductions are basically the "inverse" reductions of the ones described above. We briefly describe these reductions next.

The reduction from WSAT$^+$ on circuits of genus $N^{\Omega(1)}$ to RED-BLUE DOMINATING SET corresponds with every variable a red vertex, and with every critical gate a blue vertex; an edge exists between a red vertex and a blue vertex if and only if the variable corresponding to the red vertex is incoming to the gate corresponding to the blue vertex. The reduction from WSAT$^+$ on circuits of genus $N^{\Omega(1)}$ to HITTING SET corresponds to every variable a vertex in the hypergraph and to every gate a hyperedge containing all the vertices corresponding to the variables incoming to the gate in the circuit. The reduction from WSAT$^+$ on circuits of genus $N^{\Omega(1)}$ to SET COVER corresponds with every gate a vertex in the hypergraph, and with every variable a hyperedge containing all the vertices corresponding to the gates to which the variable is incoming. The parameter k remains unchanged in all the aforementioned reductions. It is easy to verify that the aforementioned reductions are FPT-reductions that preserve the genus of the underlying circuit. □

Theorem 5.2. *The parameterized* RED-BLUE NONBLOCKER *and* INDEPENDENT SET ON HYPERGRAPHS *problems are* FPT *on graphs/hypergraphs of genus $N^{o(1)}$ and* W[1]*-complete on graphs/hypergraphs of genus $N^{\Omega(1)}$, where N is the number of red vertices in* RED-BLUE NONBLOCKER*, and the number of vertices in* INDEPENDENT SET ON HYPERGRAPHS*.*

Proof. The FPT results follow by standard FPT-reductions that preserve the genus from RED-BLUE NONBLOCKER and INDEPENDENT SET ON HYPERGRAPHS on graphs/hypergraphs of genus $N^{o(1)}$ to WSAT$^-$ combined with Theorem 3.6. We briefly sketch these standard reductions, and refer the reader to [12, 13] for more details about these reductions.

The reduction from RED-BLUE NONBLOCKER on graphs of genus $N^{o(1)}$ to WSAT$^-$ corresponds to every red vertex in the bipartite graph a variable and to every blue vertex a critical (OR) gate; an edge is drawn between a variable and a gate if and only if an edge exists between the corresponding vertices in the graph. The reduction from INDEPENDENT SET ON HYPERGRAPHS on hypergraphs of genus $N^{o(1)}$ to WSAT$^-$ corresponds to every vertex in the hypergraph a variable and to every hyperedge a critical (OR) gate; an edge is drawn between a variable and a gate if and only if the vertex corresponding to the variable is contained in the hyperedge corresponding to the gate. The parameter k remains unchanged in all the aforementioned reductions. It is easy to verify that the aforementioned reductions are FPT-reductions that preserve the genus of the the underlying graph/hypergraph.

The W[1]-hardness results follow by standard FPT-reductions that preserve the genus from WSAT$^-$ on circuits of genus $N^{o(1)}$ to RED-BLUE NONBLOCKER and INDEPENDENT SET ON HYPERGRAPHS combined with Theorem 3.6. We briefly sketch these standard reductions.

The reduction from WSAT$^-$ on circuits of genus $N^{\Omega(1)}$ to RED-BLUE NONBLOCKER corresponds with every variable a red vertex, and with every critical gate a blue vertex; an edge exists between a red vertex and a blue vertex if and only if the variable corresponding to the red vertex is incoming to the gate corresponding to the blue vertex. The reduction from WSAT$^-$ on circuits of genus $N^{\Omega(1)}$ to INDEPENDENT SET ON HYPERGRAPHS corresponds to every variable a vertex in the hypergraph and to every gate a hyperedge containing all the vertices corresponding to the variables incoming to the gate in the circuit. The parameter k remains unchanged in all the aforementioned reductions. It is easy to verify that the aforementioned reductions are FPT-reductions that preserve the genus of the the underlying graph/hypergraph. □

6 Concluding remarks

In this paper we tried to characterize the parameterized complexity of the canonical monotone and antimonotone normalized WSAT$[t]$ problems in terms of the genus of the circuit. For WSAT$^{-}[t]$, the characterization we provided is precise. For WSAT$^{+}[t]$, however, there is still a gap between the two genus bounds of $o(\log^{2/3}(n))$ and $n^{o(1)}$. Closing this gap, or even reducing it, is a very interesting question that we leave open.

Similar characterizations of the subexponential-time computability of WSAT$^{-}[t]$ and WSAT$^{+}[t]$ in terms of the genus can be obtained. It is not difficult to prove by combining some results in this paper with a standard divide-and-conquer approach based on the separator theorem in [11], that WSAT$^{-}[t]$ and WSAT$^{+}[t]$ are solvable in subexponential-time if the genus is $o(n)$, and that they are not solvable in subexponential-time if the genus is $O(n)$ unless the exponential-time hypothesis (ETH) fails. We refer the reader to [8] for examples of how this standard approach can be applied to obtain such subexponential-time computability results. It would be interesting to see if any characterization of the approximation of the optimization versions of WSAT$^{-}[t]$ and WSAT$^{+}[t]$ based on the genus of the circuit can be derived. We also leave this as an open question.

7 Acknowledgment

The authors are grateful to Ioan Todinca for his helpful suggestions that improved the paper.

References

[1] K. Abrahamson, R. Downey, and M. Fellows. Fixed-parameter tractability and completeness IV: On completeness for W[P] and PSPACE analogues. *Annals of Pure and Applied Logic*, 73(3):235–276, 1995.

[2] D. Barrington, C. Lu, P. Miltersen, and S. Skyum. On monotone planar circuits. In *Proceedings of the Fourteenth Annual IEEE Conference on Computational Complexity*, pages 24 –31, 1999.

[3] H. Bodlaender, P. Drange, M. Dregi, F. Fomin, D. Lokshtanov, and M. Polopczuk. A $O(c^k n)$ 5-approximation algorithm for treewidth. In *Proceedings of the 50th Annual IEEE Symposium on Foundations of Computer Science*, 2013.

[4] H. Bodlaender, F. Fomin, D. Lokshtanov, E. Penninkx, S. Saurabh, and D. Thilikos. (Meta) kernelization. In *Proceedings of the 50th Annual IEEE Symposium on Foundations of Computer Science*, pages 629–638, 2009.

[5] H. Bodlaender, J. Gilbert, H. Hafsteinsson, and T. Kloks. Approximating treewidth, pathwidth, frontsize, and shortest elimination tree. *Journal of Algorithms*, 18(2):238–255, 1995.

[6] L. Cai, M. Fellows, D. Juedes, and F. Rosamond. The complexity of polynomial-time approximation. *Theory of Computing Systems*, 41(3):459–477, 2007.

[7] J. Chen, X. Huang, I. Kanj, and G. Xia. Polynomial time approximation schemes and parameterized complexity. *Discrete Applied Mathematics*, 155(2):180–193, 2007.

[8] J. Chen, I. Kanj, L. Perkovic, E. Sedgwick, and G. Xia. Genus characterizes the complexity of certain graph problems: Some tight results. *Journal of Computer and System Sciences*, 73(6):892–907, 2007.

[9] E. Demaine, F. Fomin, M. Hajiaghayi, and D. Thilikos. Subexponential parameterized algorithms on bounded-genus graphs and H-minor-free graphs. *J. ACM*, 52:866–893, 2005.

[10] E. Demaine and M. Hajiaghayi. Bidimensionality: new connections between FPT algorithms and PTASs. In *Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 590–601, 2005.

[11] H. Djidjev and S. Venkatesan. Planarization of graphs embedded on surfaces. In *Proceedings of the 21st International Workshop on Graph-Theoretic Concepts in Computer Science*, volume 1017 of *Lecture Notes in Computer Science*, pages 62–72. Springer, 1995.

[12] R. Downey and M. Fellows. *Parameterized Complexity.* Springer, New York, 1999.

[13] J. Flüm and M. Grohe. *Parameterized Complexity Theory.* Springer-verlag, Berlin, Germany, 2010.

[14] F. Fomin, P. Golovach, and D. Thilikos. Contraction obstructions for treewidth. *Journal of Combinatorial Theory (B)*, 101(5):302–314, 2011.

[15] F. Fomin, D. Lokshtanov, V. Raman, and S. Saurabh. Bidimensionality and EPTAS. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 748–759, 2011.

[16] F. Fomin, D. Lokshtanov, S. Saurabh, and D. Thilikos. Bidimensionality and kernels. In *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 503–510, 2010.

[17] J. Gross and T. Tucker. *Topological graph theory.* Wiley-Interscience, New York, NY, USA, 1987.

[18] I. Kanj, M. Pelsmajer, M. Schaefer, and G. Xia. On the induced matching problem. *Journal of Computers and System Sciences*, 77(6):1058–1070, 2011.

[19] S. Khanna and R. Motwani. Towards a syntactic characterization of PTAS. In *Proceedings of the 28th Annual ACM Symposium on Theory of Computing*, pages 468–477, 1996.

[20] T. Kloks. *Treewidth, Computations and Approximations*, volume 842 of *Lecture Notes in Computer Science*. Springer, 1994.

[21] N. Limaye, M. Mahajan, and J. Sarma. Upper bounds for monotone planar circuit value and variants. *Computational Complexity*, 18:377–412, 2009.

[22] R. Lipton and R. Tarjan. Applications of a planar separator theorem. *SIAM Journal on Computing*, 9(3):615–627, 1980.

[23] D. Marx. Completely inapproximable monotone and antimonotone parameterized problems. *J. Comput. Syst. Sci.*, 79(1):144–151, 2013.

[24] J. Savage. Planar circuit complexity and the performance of VLSI algorithms. Technical Report RR-0077, INRIA, May 1981.

[25] S. Szeider. On fixed-parameter tractable parameterizations of sat. In *The International Conferences on Theory and Applications of Satisfiability Testing*, volume 2919 of *Lecture Notes in Computer Science*, pages 188–202, 2004.

[26] G. Turán. On the complexity of planar boolean circuits. *Computational Complexity*, 5:24–42, 1995.

[27] J. van Rooij, H. Bodlaender, and P. Rossmanith. Dynamic programming on tree decompositions using generalised fast subset convolution. In *Proceedings of the 17th Annual European Symposium on Algorithms*, volume 5757 of *Lecture Notes in Computer Science*, pages 566–577. Springer, 2009.

[28] I. Wegener. *The complexity of Boolean functions.* Wiley-Teubner, 1987.

[29] D. West. *Introduction to graph theory.* Prentice Hall Inc., Upper Saddle River, NJ, 1996.