Parameterized Complexity and Subexponential-Time Computability*

Jianer Chen[1],** and Iyad A. Kanj[2],***

[1] Department of Computer Science and Engineering, Texas A&M University, College Station, TX 77843
chen@cs.tamu.edu

[2] School of Computing, DePaul University, 243 S. Wabash Avenue, Chicago, IL 60604-2301
ikanj@cs.depaul.edu

Abstract. Since its inception in the 1990's, parameterized complexity has established itself as one of the major research areas in theoretical computer science. Parameterized and kernelization algorithms have proved to be very useful for solving important problems in various domains of science and technology. Moreover, parameterized complexity has shown deep connections to traditional areas of theoretical computer science, such as structural complexity theory and approximation algorithms.

In this paper, we discuss some of the recent results pertaining to the relation between parameterized complexity and subexponential-time computability. We focus our attention on satisfiability problems because they play a key role in the definition of both parameterized complexity and structural complexity classes, and because they model numerous important problems in computer science.

1 Introduction

Parameterized complexity was established in the early 1990's by the seminal work of Downey and Fellows. It was instigated by the demands of real-world applications, and by the belief that computational complexity should "serve the community," and should be "used not only in the pursuit of the declared objectives but also in the design of heuristic and approximation algorithms for

* This paper is dedicated to the 60th birthday of Michael R. Fellows. Many of the results examined in this paper were authored or co-authored by Michael, and those that were not, would probably never have existed without his efforts. If parameterized complexity would not have started without Michael, Rodney, and two bottles of 1989 Villa Maria Merlot/Cabarnet Sauvignon, then it definitely would not have flourished and matured into such an important area of theoretical computer science without Michael's great ideas, efforts, and inspirations.

** This work was supported in part by the USA National Science Foundation under the grants CCF-0830455 and CCF-0917288.

*** Supported in part by a DePaul University Competitive Research Grant.

problems that are hard, but for which, nevertheless, something must be done" [24]. Today, applications of efficient parameterized and kernelization algorithms for solving important problems that are otherwise intractable from the traditional complexity theory perspective, are prevalent. Parameterized complexity has matured into a very exciting research area of theoretical computer science, with applications spanning numerous domains of science and technology.

The rich positive toolkit of novel techniques for designing efficient parameterized and kernelization algorithms was accompanied by a corresponding "negative" toolkit that supports a theory of parameterized intractability. While studying the parameterized intractability of the DOMINATING SET problem [24], Downey and Fellows sharply observed that there is a rich structure in parameterized intractability theory. A parameterized intractability hierarchy, the W-hierarchy, was subsequently introduced to classify the level of intrinsic intractability of parameterized problems [24]. Research in the last twenty years revealed that the W-hierarchy provides a deep structural characterization of parameterized complexity [24]. In addition, the theory is remarkably applicable to a wide range of natural computational problems. This motivated researchers in theoretical computer science, and in parameterized complexity in particular, to investigate the structural relation between parameterized complexity and traditional areas of theoretical computer science (e.g., structural complexity, approximation, etc).

Perhaps one of the most important problems that has deep roots in the aforementioned areas is the satisfiability problem, including all its variants. The currently-best algorithms for satisfiability problems are essentially based on brute-force methods that enumerate all possible solutions, which obviously requires exponential time. It has become clear that the existence of faster exponential-time algorithms for satisfiability problems is closely related to the computational intractability of a large class of well-known NP-hard problems, measured from a number of different angles, such as computational time and space, fixed-parameter tractability, and approximation. For example, Impagliazzo, Paturi, and Zane showed that the subexponential-time computability of the 3-SAT problem is equivalent to the subexponential time computability of a large class of well-known NP-hard problems [31]; this class is closed under subexponential-time preserving reductions, called *serf-reductions*. This led researchers in theoretical computer science to formulate a hypothesis, which became known as the *exponential-time hypothesis*, shortly ETH, conjecturing that no member in this class is solvable in subexponential time.

The subexponential-time computability of weighted satisfiability on bounded depth circuits is closely related to the fixed-parameter tractability of the W-hierarchy in parameterized complexity theory (see for instance [1, 9, 12, 14, 19–21, 23, 24]). Research on parameterized complexity revealed more subtle relations between the computational complexity of NP-hard problems and the (sub)exponential-time computability of satisfiability problems [9, 12, 14, 33, 39]. For example, it is now known that efficient polynomial-time approximation schemes of a number of NP-hard problems, and parameterized algorithms that are asymptotically more efficient than the brute-force enumeration for many W-hard problems, are all dependent of the subexponential-time computability of various satisfiability problems (see for instance [9, 12, 14, 33, 39]). In particular, due to the aforementioned research, it is now known that the classification of parameterized intractability has a correspondence to the exact computability of satisfiability problems on various circuit families. This line of research deepened our understanding of the structural relation between the two computational frameworks of parameterized complexity and subexponential-time computability, and resulted in new tools for deriving computational lower bounds on parameterized computation, exact computation, and approximation algorithms.

In the current paper, we discuss some of the results related to the relation between the parameterized intractability and the computational complexity of a variety of satisfiability problems. This relation is not surprising, given that the W-hierarchy in parameterized complexity was defined mainly based on weighted satisfiability problems. Nevertheless, the close connection between these two different research frameworks has not been rigorously studied until very recently.

The systematic research in this direction started with the results of Abrahamson, Downey, and Fellows [1], and Downey and Fellows [24]. In [1], it was shown that $W[P] = FPT$ ($W[P]$ is the parameterized complexity class characterized by the weighted satisfiability problem restricted to polynomial-size Boolean circuits) implies that CIRCUIT SATISFIABILITY (satisfiability of polynomial-size circuits) is computable in subexponential time. It was also shown in [1] that, for any even $t \geq 1$, the collapse of the W-hierarchy at its t-th level (i.e., $W[t] = FPT$) implies that SAT$[t]$ (t-level satisfiability) is solvable in time $2^{o(n)}m^{O(1)}$ (m is the instance size).[1] This later result was refined by Downey and Fellows [24] to all levels of the W-hierarchy. Downey and Fellows [24] showed that: (1) for any $t \geq 2$, $W[t] = FPT$ implies that SAT$[t]$ is computable in subexponential time; and (2) $W[1] = FPT$ implies that ETH fails, which subsequently implies the subexponential-time computability of several well-known problems including 3-SAT, INDEPENDENT SET, and VERTEX COVER. Those results were further refined and extended in [12, 14], where it was shown that the condition $W[t] = FPT$ in (1) and (2) can be relaxed (see Section 4), and that the subexponential-time computability of SAT$[t]$, in turn, implies the collapse of the W-hierarchy at its $(t-1)$-st level ($W[t-1] = FPT$) for $t \geq 2$, and for $t = 1$ implies the failure of ETH. The previous results were exploited further in [12, 14] to derive lower bounds on the computability and the approximation of well-known NP-hard problems, such as INDEPENDENT SET, CLIQUE, DOMINATING SET, based on parameterized complexity hypotheses. Important questions along this line of research remained open however, including the following: What is the equivalent, from the parameterized complexity perspective, of the subexponential-time computability of various satisfiability problems?

[1] The $o(\cdot)$ notation in this paper denotes the $o^{\text{eff}}(\cdot)$ notation (see, for instance, [26]). More formally, 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 $n \geq n_0$.

Downey et al. [23] were the first to try to answer this question. They defined a parameterized complexity class, called $M[1]$, comprised between FPT and $W[1]$, consisting of a "miniaturization" of weighted circuit satisfiability, and showed that $M[1] = FPT$ is equivalent to ETH fails. The idea of a miniaturization of a problem was explored earlier in the work of Abrahamson et al. [1], and Cai and Juedes [9], and Downey et al. [23] provided a formal definition for this notion and studied it systematically. Flum and Grohe [25], Chen and Flum [19, 20], and Chen and Grohe [21], launched a systematic study of the relation between parameterized complexity and subexponential-time computability, using the notion of miniaturization. Chen and Grohe [21] were able to give a correspondence between the subexponential-time computability of certain satisfiability problems and parameterized complexity classes.

We will focus on some of the key results pertaining to the relation between parameterized complexity and subexponential-time computability, and their applications. We will also try to describe some of the problems that remain open in this line of research. While we tried our best to include most of the recent results on to these topics, we do apologize in advance for any relevant result that we may have omitted; certainly, this was not our intention.

Before we close this section, we would like to mention a recent breakthrough-result in complexity theory by Williams [42], who proved that the non-uniform ACC class does not contain NTIME$[2^n]$, the class of languages that are solvable in nondeterministic time $O(2^n)$. This is regarded as a very significant advance in complexity theory, as it was even unknown whether the class EXP^{NP} is contained in a weaker ACC class of languages accepted by circuit families of polynomial-size and depth 3 with more restricted modular gates. There are at least two directions in which Williams' result is relevant to parameterized complexity and subexponential-time computability. First, a major component of Williams' approach is faster exact algorithms for satisfiability problems. In particular, Williams developed a $2^{n-\Omega(n^\delta)}$-time algorithm for the satisfiability problem on subexponential-size ACC-circuits, where δ is a constant dependent on the circuit depth. Second, the computation model considered, i.e., ACC-circuits, is closely related to the generic complete problems for the W-hierarchy, i.e., the weighted satisfiability on bounded depth Boolean circuits [24].

2 Preliminaries

A *circuit* is a directed acyclic graph. The nodes of in-degree 0 are called *inputs*, and are labeled either by *positive literals* x_i or by *negative literals* $\overline{x}_i$. The nodes of in-degree larger than 0 are called *gates* and are labeled with Boolean operators AND or OR. A special gate of out-degree 0 is designated as the *output* node. 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 node to the output gate of the circuit. A circuit represents a Boolean function in a natural way. Using the results in [11], every circuit can be re-structured into an equivalent circuit with the same monotonicity and number of input variables, same depth, and such that all inputs are in level 0, all AND and OR gates are organized into alternating levels with edges only going from a level to the next level, and with at most a polynomial increase in the circuit size. Thus, without loss of generality, we will implicitly assume that circuits are in this leveled form. A circuit is a *Π-circuit* if its output gate is an AND gate, and is a *Π_h-circuit* if it is a Π-circuit of depth h. We say that a truth assignment τ to the input 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. A *CNF formula* is a conjunction of a set of *clauses* where each clause is a disjunction of literals. For a CNF formula F with n input variables, we can naturally correspond an equivalent Π_2-circuit C_F with n input variables.

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* [24] if there is a (parameterized) algorithm that decides whether an input (x, k) is a member of Q in time $f(k)|x|^c$, where c is a fixed constant and $f(k)$ is a computable function independent of the input length $|x|$. Let FPT denote the class of all fixed-parameter tractable parameterized problems.

The Π_t-CIRCUIT SATISFIABILITY problem where $t \geq 2$, abbreviated SAT$[t]$ henceforth, is defined as follows: Given a Π_t-circuit C, decide if C is satisfiable. For instance, the SAT[2] problem is the same as the satisfiability problem on CNF formulas (CNF-SAT). We will also study the parameterized problems based on the "weighted version" of the satisfiability problems on circuits. In particular, for $t \geq 2$, the WEIGHTED Π_t-CIRCUIT SATISFIABILITY problem, abbreviated WCS$[t]$ is for a given Π_t-circuit C and a given parameter k, to decide if C has a satisfying assignment of weight k. Similarly, the WEIGHTED MONOTONE Π_t-CIRCUIT SATISFIABILITY problem, abbreviated WCS$^+[t]$, and the WEIGHTED ANTIMONOTONE Π_t-CIRCUIT SATISFIABILITY problem, abbreviated WCS$^-[t]$ are the WCS$[t]$ problems on, respectively, monotone circuits and antimonotone circuits. We denote by WCNF 2-SAT$^-$ the WCS$^-$[2] problem with the restriction that the fan-in of each gate at level 1 of the input circuit is bounded by 2. Equivalently, each instance of WCNF 2-SAT$^-$ consists of a parameter k and a CNF formula in which all literals are negative and each clause is a disjunction of at most two literals. Finally, let 3-SAT be the CNF-SAT problem with the restriction that each clause in the input formula is a disjunction of at least 3 literals.

The optimization class SNP introduced by Papadimitriou and Yannakakis [38] consists of all search problems expressible by second-order existential formulas whose first-order part is universal. Impagliazzo and Paturi [31] introduced the notion of *completeness* for the class SNP under *serf-reductions*, and identified a class of problems which are complete for SNP under serf-reductions, such that the subexponential-time computability for any of these problems implies the subexponential-time computability of all problems in SNP. Many well-known NP-hard problems are proved to be complete for SNP under the serf-reduction, including 3-SAT, VERTEX COVER, and INDEPENDENT SET, for which extensive efforts have been made in the last three decades to develop subexponential-time algorithms with no success [43]. This fact has led to the *exponential-time hypothesis*, ETH, which is equivalent to the statement that not all SNP problems are solvable in subexponential-time:

> *Exponential-Time Hypothesis* (ETH): The problem 3-SAT cannot be solved in time $2^{o(n)}$, where n is the number of variables in the input formula.

The ETH has become a standard hypothesis in the area of parameterized algorithms and complexity and of exact algorithms and complexity.

To study the fixed-parameter tractability, the *fpt-reduction* has been introduced [24]: a parameterized problem Q is *fpt-reducible* to a parameterized problem Q' if there is an algorithm M that transforms each instance (x, k) of Q into an instance $(x', g(k))$ (g is a function of k only) of Q' in time $f(k)|x|^c$, where f and g are computable functions and c is a constant, such that $(x, k) \in Q$ if and only if $(x', g(k)) \in Q'$.

Based on the notion of fpt-reducibility, a hierarchy of parameterized complexity, the *W-hierarchy*, has been introduced. At the 0-th level of the hierarchy lies the class FPT, and at the i-th level for $i > 0$, the class $W[i]$. The original definition of the $W[i]$ classes was based on the notion of the *weft* of a circuit, which is the maximum number of "large gates" (i.e., gates whose fan-in is larger than a prespecified constant) on any path from an input gate to the output gate of the circuit [24]. The previous definition, however, was shown to be equivalent to the following definition by Downey and Fellows (see [24]), which we use in this paper. The class $W[1]$ consists of all parameterized problems that are fpt-reducible to the problem WCNF 2-SAT$^-$. For an even $t \geq 2$, the class $W[t]$ consists of all parameterized problems that are fpt-reducible to the problem WCS$^+[t]$, and for an odd $t \geq 3$, the class $W[t]$ consists of all parameterized problems that are fpt-reducible to the problem WCS$^-[t]$. To simplify our statements, we will denote by WCS$^*[t]$ the problem WCS$^+[t]$ if t is even and the problem WCS$^-[t]$ if t is odd. Therefore, for all $t \geq 2$, the class $W[t]$ consists of all parameterized problems that are fpt-reducible to the problem WCS$^*[t]$.

A parameterized problem Q is *$W[i]$-hard* if every problem in $W[i]$ is fpt-reducible to Q, and is *$W[i]$-complete* if in addition Q is in $W[i]$. By the definition, the problem WCNF 2-SAT$^-$ is $W[1]$-complete, and for all $t \geq 2$, the problem WCS$^*[t]$ is $W[t]$-complete. If any $W[i]$-hard problem is in FPT, then $W[i] = FPT$, which, to the common belief of researchers in parameterized complexity, is very unlikely [24].

We have the following relation among parameterized complexity classes [24]:

$$FPT \subseteq W[1] \subseteq W[2] \subseteq \ldots$$

For more information about parameterized complexity we refer the reader to [24, 26, 36].

3 ETH, *W*[1], and CNF-SAT

In this section we discuss some of the results on the relation between the parameterized complexity and the subexponential-time computability of important satisfiability problems on Π_2-circuits, and their consequences on the computability of some natural NP-hard problems. We start with the following folklore results that can be easily verified by the reader (see for example [9]):

Lemma 1. *If a parameterized problem is solvable in time $2^{o(k \log n)} n^{O(1)}$, where n is the input length and k is the parameter, then the problem is fixed-parameter tractable.*

Fact 31. *The function $(c \log n)^{O(k)}$ is bounded above by $f(k)n^{O(1)}$, where f is a function of k only.*

The following result is a consequence of a result in [9]:

Theorem 1. *If* CNF-SAT *is solvable in time $2^{o(n)} m^{O(1)}$, then $W[1] = FPT$, where n is the number of variables and m is the formula size.*

Proof. Suppose that CNF-SAT is solvable in time $2^{o(n)} m^{O(1)}$. We consider the WCNF 2-SAT$^-$ problem. Since this problem is complete for $W[1]$, it suffices to show that it can be solved in time $f(k) m_1^{O(1)}$ where f is a function independent of the circuit size m_1. Note that since $m_1 = n^{O(1)}$, where n is the number of variables in the circuit (each gate at level-1 has fan-in at most 2), the problem reduces to showing that WCNF 2-SAT$^-$ is solvable in time $f(k)n^{O(1)}$.

Let (C, k) be an instance of WCNF 2-SAT$^-$, and note that the gates at level 1 in C are OR gates, each of fan-in at most two. Let $\overline{x}_1, \ldots, \overline{x}_n$ be the input literals to C. We will construct a circuit C' from C with $k\lceil \log n \rceil$ input variables, such that C has a weight-k assignment if and only if C' is satisfiable. The input variables in C' are divided into k blocks $B_1, \ldots, B_k$, where block B_i, $i = 1, \ldots, k$, consists of $r = \lceil \log n \rceil$ input variables $z_i^1, \ldots, z_i^r$. Also, for every input variable z_i^j, $i \in \{1, \ldots, k\}$, $j \in \{1, \ldots, r\}$, we associate the input literal $\overline{z}_i^j$ to denote its negation. Informally speaking, each block B_i will contain the encoding of an input variable whose value is 1 in a weight-k assignment to C. We show how to connect the new input variables and their negations to the level-1 OR gates in C. Let g be a gate at level-1 in C, and suppose that $\overline{x}_p, \overline{x}_q$, are connected to g. (We assume that g has fan-in exactly two as the case when g has fan-in 1 is much easier to handle.) Now $\overline{x}_p$ is 1 if and only if x_p is 0, if and only if none of the blocks B_i, $i = 1, \ldots, k$ contains the binary representation of p. Thus, in C' we will connect the new input variables to g as follows. We introduce k new OR gates $g_p^1, \ldots, g_p^k$. Each gate g_p^i, $i = 1, \ldots, k$, has exactly r inputs, and its input comes only from input variables in block B_i and their negations. Informally speaking, each gate g_p^i will be satisfied if and only if block B_i does not contain the binary representation of p, and hence, does not encode x_p. Suppose that the binary representation of p is $b_1 b_2 \ldots b_r$. For $i = 1, \ldots, k$, the input to g_p^i is determined as follows. For $j = 1, \ldots, r$, if $b_j = 0$, then connect z_i^j to g_p^i, and if $b_j = 1$, then connect $\overline{z}_i^j$ to g_p^i. Now replace the connection from $\overline{x}_p$ to g by the connections from all gates g_p^i, $i = 1, \ldots, k$ to an AND gate g_p which feeds into g. We do the same for $\overline{x}_q$. Now gate g is equivalent to $g_p \vee g_q$, where $g_p = \bigwedge_{i=1}^{k} g_p^i$ and $g_p = \bigwedge_{i=1}^{k} g_q^i$. By the distributive law, we can write $g = \bigwedge_{i,j=1,\ldots,k} (g_p^i \vee g_q^i)$, and the AND gate in g can be merged with the output AND gate of the circuit C. We repeat the above construction for every level-1 gate in C. Since the g_p^i's and the g_q^i's for every level-1 gate g are OR gates, the resulting circuit is a two-level circuit, where the top level consists of OR gates that all feed into the single output AND gate of C.

Now we can add enforcement circuitry to ensure that the k blocks encode k distinct input variables. This can be simply achieved by adding a circuitry consisting of $\binom{k}{2}$ subcircuits, each subcircuit enforces that the two blocks that feed into it are distinct. To do so, each subcircuit performs a bitwise XOR operation to the corresponding variables in the two blocks. Since the number of input variables to each subcircuit is $O(\log n)$, each subcircuit can be transformed into a subcircuit in the CNF form in $n^{O(1)}$ time, whose output AND gate can then be merged with the output AND gate of C.

Let C' be the resulting circuit. Clearly, C' has size $n^{O(1)}$ and can be constructed in $n^{O(1)}$ time. Moreover, from the above discussion, we know that C' consists of two levels, where the top level consists only of OR gates, and the bottom level consists of the output AND gate of the circuit. Since the k input blocks in C' basically encode the k input variables in C with value 1 in a weight-k assignment to C, it is not difficult to verify that C has a weight-k truth assignment if and only if C' is satisfiable. Now C' is an instance of CNF-SAT with $k \cdot r$ input variables. It follows that we can decide if C' is satisfiable in time bounded by $2^{o(k \log n)} n^{O(1)}$. By Lemma 1, it follows that WCNF 2-SAT$^-$ is fixed-parameter tractable.

Using reductions to problem kernel, and some standard self reductions, Cai and Juedes [9] were able to preclude the existence of subexponential-time parameterized algorithms for several problems under the assumption that ETH holds (n is the instance size, and k is the natural parameter in the corresponding problem):

Theorem 2 ([9]). *The following problems can be solved in time $2^{o(k)} n^{O(1)}$ if and only if ETH fails:* VERTEX COVER, MAX h-SAT, Δ-VERTEX COVER *(the graph has degree bounded by the constant Δ),* Δ-INDEPENDENT SET, *and* Δ-DOMINATING SET.

Note that all the above problems can be solved in time $2^{O(k)} n^{O(1)}$. So assuming ETH, the above theorem rules out the existence of significantly-better parameterized algorithms for those problems than the ones that are currently known.

Using kernelization, planarization techniques, and standard reductions, Cai and Juedes [9] were able to extend the above lower bound results to planar graphs:

Theorem 3 ([9]). *Unless ETH fails, the following problems cannot be solved in time* $2^{o(\sqrt{k})}n^{O(1)}$*:* PLANAR VERTEX COVER*,* PLANAR INDEPENDENT SET*,* PLANAR DOMINATING SET*, and* PLANAR RED/BLUE DOMINATING SET*.*

Some of the results in the previous theorem can also be extended to bounded-degree planar graphs [17] to obtain the same lower bounds.

The following theorem can be viewed as providing a partial converse to Theorem 1, after noting that the statement that ETH fails is equivalent to subexponential-time computability of 3-SAT. This result is due to Downey and Fellows [24]. The proof given here, which appears in [12], is different than the original proof, since the original proof was a corollary of a more general result.

Theorem 4. *If* $W[1] = FPT$ *then the hypothesis ETH fails.*

Proof. Since INDEPENDENT SET is W[1]-complete under the fpt-reduction [24] and VERTEX COVER is complete for SNP under serf-reductions [31], it suffices to show that if INDEPENDENT SET is solvable in time $f(k)n^{O(1)}$ then VERTEX COVER is solvable in time $2^{o(k)}n^{O(1)}$. Assume that there is an algorithm A which determines whether there exists an independent set of size k in a graph G with n vertices in $f(k)n^{O(1)}$ time. We will show that the VERTEX COVER problem can be solved in time $2^{o(k)}n^{O(1)}$. Without loss of generality, we can assume that the function f is nondecreasing, unbounded, and that $f(k) \geq 2^k$. Define f^{-1} by $f^{-1}(h) = \max\{q \mid f(q) \leq h\}$. Since the function f is nondecreasing and unbounded, the function f^{-1} is also nondecreasing and unbounded, and satisfies $f(f^{-1}(h)) \leq h$. From $f(k) \geq 2^k$, we have $f^{-1}(h) \leq \log h$.

Let $(G = (V, E), k)$ be an instance of VERTEX COVER. By the kernelization result for VERTEX COVER [16], we can assume that G has at most $n \leq 2k$ vertices. We partition the n vertices of G into $k' = \lfloor f^{-1}(k) \rfloor$ blocks $B_1, B_2, \ldots, B_{k'}$ each of size at most $\lceil \frac{n}{\lfloor f^{-1}(k) \rfloor} \rceil$. (Without loss of generality, we shall assume that $\lfloor f^{-1}(k) \rfloor \geq 1$.) Observe that G has a vertex cover of size k if and only if there exists a way to partition k into $k_1, \ldots, k_{k'}$ (i.e., $k = k_1 + k_2 + \cdots + k_{k'}$), and there are subsets $V_i' \subseteq B_i$, $i = 1, \ldots, k'$ with $|V_i'| = k_i$, such that $\bigcup_{i=1}^{k'} V_i'$ is a vertex cover for G. Since $|B_i| \leq \lceil \frac{n}{\lfloor f^{-1}(k) \rfloor} \rceil$, this approach converts the single question "does G have a vertex cover of size k?" into at most

$$\begin{aligned}
(\lceil \frac{n}{\lfloor f^{-1}(k) \rfloor} \rceil)^{k'} &\leq (\lceil \frac{2k}{\lfloor f^{-1}(k) \rfloor} \rceil)^{\lfloor f^{-1}(k) \rfloor} \\
&\leq (2k)^{f^{-1}(k)} \\
&\leq 2^{\log(2k) \cdot f^{-1}(k)} \\
&\leq 2^{\log(2k) \cdot \log k} = 2^{o(k)}
\end{aligned}$$

more restrictive questions of the type "does G have a vertex cover V' of size $k = k_1 + k_2 + \cdots + k_{k'}$ with $|B_i \cap V'| = k_i$?". Hence, we can determine whether G has a vertex cover of size k by answering at most $2^{o(k)}$ questions individually.

To answer each of the $2^{o(k)}$ questions, we use the algorithm A for INDEPENDENT SET. Given G, k, and $k_1, \ldots, k_{k'}$ such that $k = k_1 + k_2 + \cdots + k_{k'}$, we construct a graph $G^* = (V^*, E^*)$ as follows. For each block of vertices B_i in G, and for each subset $B_{ij} \subseteq B_i$ with $|B_{ij}| = k_i$, add a vertex v_{ij} to V^* if B_{ij} is a vertex cover of $G(B_i)$ (the subgraph of G induced by B_i). Add edges to E^* so that the collection of the vertices v_{ij} associated with block B_i, $i = 1, \ldots, k'$, forms a clique. In addition, for each v_{ij}, $v_{kl} \in V^*$, where $i \neq k$, add the edge (v_{ij}, v_{kl}) to E^* if $B_{ij} \cup B_{kl}$ does not form a vertex cover for $G(B_i \cup B_k)$. This completes the construction of G^*. To determine if G has a vertex cover of size k with the properties mentioned above, it suffices to use algorithm A to determine if G^* has an independent set of size k'. We prove the correctness of this claim.

Assume that G^* has an independent set I of size k'. Since G^* has k' disjoint cliques, exactly one vertex from each set $V_i^* = \{v_{ij} \mid v_{ij} \in V^*\}$ is in I. Let $V' = \cup_{v_{ij} \in I} B_{ij}$. Since $|B_{ij}| = k_i$, and at most one B_{ij} is included in V', it follows that $|V' \cap B_i| = k_i$, and $|V'| = k$. Thus, it suffices to prove that V' is a vertex cover of G. Let $(u, v) \in E$, and let $u \in B_i$ and $v \in B_k$. If $i = k$, then it must be the case that either u or $v \in V'$. To see this, note that there exists a $v_{ij} \in I \subseteq V^*$, which means that $B_{ij} \subseteq V'$ by the definition of V'. Since $v_{ij} \in V^*$, B_{ij} is a vertex cover of $G(B_i)$, and either u or v must be in $B_{ij} \subseteq V'$. Suppose now that $i \neq k$, and let v_{ij}, v_{kl} be the two vertices in V_i^* and V_j^*, respectively, that are in I. Then it must be the case that $u \in B_{ij}$ or $v \in B_{kl}$, otherwise $B_{ij} \cup B_{kl}$ is not a vertex cover of $G(B_i \cup B_k)$, which would imply that there is an edge between v_{ij} and v_{kl} in G^*, contradicting the fact that I is an independent set of G^*. It follows that either u or v is in V'. This shows that V' is a vertex cover of G. To prove the converse, assume that G has a vertex cover V' of size $k = k_1 + k_2 + \cdots + k_{k'}$ with $|B_i \cap V'| = k_i$. Let $I = \{v_{ij} \mid B_{ij} = B_i \cap V'\}$. It is clear that $I \subseteq V^*$ and $|I| = k'$, since for each i, B_{ij} has k_i vertices and it is a vertex cover of $G(B_i)$. Furthermore, I is an independent set in G^* because for each v_{ij}, $v_{kl} \in I$, $(v_{ij}, v_{kl}) \notin E^*$. This is true since $B_{ij} \cup B_{kl} = V' \cap (B_i \cup B_k)$ is a vertex cover of $G(B_i \cup B_k)$.

Therefore, we can use algorithm A to determine whether G has a vertex cover V' of size $k = k_1 + k_2 + \cdots + k_{k'}$, by checking whether G^* has an independent set I of size k'. The graph G^* has at most $N = 2^{\lceil \frac{2k}{\lfloor f^{-1}(k) \rfloor} \rceil} \cdot k' \leq 2^{\lceil \frac{2k}{\lfloor f^{-1}(k) \rfloor} \rceil} \cdot f^{-1}(k) = 2^{o(k)}$ vertices because $|B_i| \leq \lceil \frac{2k}{\lfloor f^{-1}(k) \rfloor} \rceil$, and there are at most $2^{|B_i|}$ possible subsets B_{ij} of size k_i. Therefore, the time taken by applying the algorithm A to the instance (G^*, k') is of the order

$$f(k')N^{O(1)} \leq f(f^{-1}(k))N^{O(1)} \leq k \cdot N^{O(1)} = 2^{o(k)} n^{O(1)}$$

after observing that $N^{O(1)} = 2^{o(k)}$. Noting that the time needed to construct G^* is $N^{O(1)} = 2^{o(k)}$, and that applying the kernelization algorithm for VERTEX COVER takes polynomial time in n, it follows that the VERTEX COVER problem can be solved in time $n^{O(1)} + 2^{o(k)} \cdot 2^{o(k)} \cdot n^{O(1)} = 2^{o(k)} n^{O(1)}$. This completes the proof.

In fact, Theorem 4 can be strengthened to the following result:

Theorem 5. *If the $W[1]$-complete problem* WCNF 2-SAT$^-$ *is solvable in time $f(k)m^{o(k)}$, where m is the size of the input formula, then the hypothesis ETH fails.*

The interested readers are referred to [12, 14] for a detailed proof.

4 A General Framework

In this section, we present two generic results that establish certain relations between the parameterized complexity of weighted satisfiability problems and the subexponential-time computability of their unweighted versions. Since weighted satisfiability problems are complete for the W-hierarchy, this leads to a relation between the subexponential-time computability of natural satisfiability problems and the collapse of the W-hierarchy. In Section 6, we will present some applications of these results to obtain computational lower bounds on the parameterized complexity and on the approximation of natural problems. The results are mainly due to the work in [12, 14], and can be viewed as generalizations and strengthening of the results in the previous section.

We start with the following lemma, which will be used in the proof of the next theorem:

Lemma 2. *Let $t \geq 2$ be an integer. There is an algorithm A_1 that, for a given integer $r > 0$, transforms each Π_t-circuit C_1 of n_1 input variables and size m_1 into an instance (C_2, k) of* WCS$^*[t]$, *where $k = \lceil n_1/r \rceil$ and the Π_t-circuit C_2 has $n_2 = 2^r k$ input variables and size $m_2 \leq 2m_1 + 2^{2r+1}k$, such that C_1 is satisfiable if and only if (C_2, k) is a yes-instance of* WCS$^*[t]$. *The running time of the algorithm A_1 is bounded by $O(m_2^2)$.*

Proof. Let $k = \lceil n_1/r \rceil$. Divide the n_1 input variables $x_1, \ldots, x_{n_1}$ of the Π_t-circuit C_1 into k blocks $B_1, \ldots, B_k$, where block B_i consists of input variables $x_{(i-1)r+1}, \ldots, x_{ir}$, for $i = 1, \ldots, k-1$, and block B_k consists of input variables $x_{(k-1)r+1}, \ldots, x_{n_1}$. Denote by $|B_i|$ the number of variables in block B_i. Then $|B_i| = r$, for $1 \leq i \leq k-1$, and $|B_k| \leq r$. For an integer j, $0 \leq j \leq 2^{|B_i|} - 1$, denote by $\text{bin}_i(j)$ the length-$|B_i|$ binary representation of j, which can also be interpreted as an assignment to the variables in block B_i.

We construct a new set of input variables in k blocks $B'_1, \ldots, B'_k$. Each block B'_i consists of $s = 2^r$ variables $z_{i,0}$, $z_{i,1}$, $\ldots$, $z_{i,s-1}$. The Π_t-circuit C_2 is constructed from the Π_t-circuit C_1 by replacing the input gates in C_1 by the new input variables in $B'_1, \ldots, B'_k$. We consider two cases.

Case 1. t is even. Then all level-1 gates in the Π_t-circuit C_1 are OR gates. We connect the new variables $z_{i,j}$ to these level-1 gates to construct the circuit C_2 as follows. Let x_q be an input variable in C_1 such that x_q is the h-th variable in block B_i. If the positive literal x_q is an input to a level-1 OR gate g_1 in C_1, then all positive literals $z_{i,j}$ in block B'_i such that $0 \leq j \leq 2^{|B_i|} - 1$ and the h-th bit in $\text{bin}_i(j)$ is 1 are connected to gate g_1 in the circuit C_2. If the negative literal

$\overline{x}_q$ is an input to a level-1 OR gate g_2 in C_1, then all positive literals $z_{i,j}$ in block B'_i such that $0 \leq j \leq 2^{|B_i|} - 1$ and the h-th bit in $\text{bin}_i(j)$ is 0 are connected to gate g_2 in the circuit C_2.

Note that if the size $|B_k|$ of the last block B_k in C_1 is smaller than r, then the above construction for block B'_k is only on the first $2^{|B_k|}$ variables in B'_k, and the last $s - 2^{|B_k|}$ variables in B'_k have no output edges, and hence become "dummy variables".

We also add an "enforcement" circuitry to the circuit C_2 to ensure that every satisfying assignment to C_2 assigns the value 1 to at least one variable in each block B'_i. This can be achieved by having an OR gate for each block B'_i, whose inputs are connected to all positive literals in block B'_i and whose output is an input to the output gate of the circuit C_2 (for block B'_k, the inputs of the OR gate are from the first $2^{|B_k|}$ variables in B'_k). This completes the construction of the circuit C_2. It is easy to see that the circuit C_2 is a monotone Π_t-circuit (note that $t \geq 2$ and hence the enforcement circuitry does not increase the depth of C_2). Thus, (C_2, k) is an instance of the problem WCS$^+[t]$.

We verify that the circuit C_1 is satisfiable if and only if the circuit C_2 has a satisfying assignment of weight k. Suppose that the circuit C_1 is satisfied by an assignment τ. Let τ_i be the restriction of τ to block B_i, $1 \leq i \leq k$. Let j_i be the integer such that $\text{bin}_i(j_i) = \tau_i$. Then according to the construction of the circuit C_2, by setting $z_{i,j_i} = 1$ and all other variables in B'_i to 0, we can satisfy all level-1 OR gates in C_2 whose corresponding level-1 OR gates in C_1 are satisfied by the assignment τ_i. Doing this for all blocks B_i, $1 \leq i \leq k$, gives a weight-k assignment τ' to the circuit C_2 that satisfies all level-1 OR gates in C_2 whose corresponding level-1 OR gates in C_1 are satisfied by τ. Since τ satisfies the circuit C_1, the weight-k assignment τ' satisfies the circuit C_2.

Conversely, suppose that the circuit C_2 is satisfied by a weight-k assignment τ'. Because of the enforcement circuitry in C_2, τ' assigns the value 1 to exactly one variable in each block B'_i (in particular, in block B'_k, this variable must be one of the first $2^{|B_k|}$ variables in B'_k). Now suppose that in block B'_i, τ' assigns the value 1 to the variable z_{i,j_i}. Then we set an assignment τ_i to the block B_i in C_1 such that $\tau_i = \text{bin}_i(j_i)$. By the construction of the circuit C_2, the level-1 OR gates satisfied by the variable $z_{i,j_i} = 1$ are all satisfied by the assignment τ_i. Therefore, if we make an assignment τ to the circuit C_1 such that the restriction of τ to block B_i is τ_i for all i, then the assignment τ will satisfy all level-1 OR gates in C_1 whose corresponding level-1 OR gates in C_2 are satisfied by τ'. Since τ' satisfies the circuit C_2, we conclude that the circuit C_1 is satisfiable.

This completes the proof that when t is even, the circuit C_1 is satisfiable if and only if the constructed pair (C_2, k) is a yes-instance of WCS$^+[t]$.

Case 2. t is odd. Then all level-1 gates in the Π_t-circuit C_1 are AND gates. We connect the new variables $z_{i,j}$ to these level-1 gates to construct the circuit C_2 as follows. Let x_q be an input variable in C_1 and be the h-th variable in block B_i. If the positive literal x_q is an input to a level-1 AND gate g_1 in C_1, then all negative literals $\overline{z}_{i,j}$ in block B'_i such that $0 \leq j \leq 2^{|B_i|} - 1$ and the h-th bit in $\text{bin}_i(j)$ is 0 are inputs to gate g_1 in C_2. If the negative literal $\overline{x}_q$ is an input to a level-1 AND gate g_2 in C_1, then all negative literals $\overline{z}_{i,j}$ in block B'_i such that $0 \le j \le 2^{|B_i|} - 1$ and the h-th bit in $\text{bin}_i(j)$ is 1 are inputs to gate g_2 in C_2.

For the last $s - 2^{|B_k|}$ variables in the last block B'_k in C_2, we connect the negative literals $\overline{z}_{k,j}$, $2^{|B_k|} \le j \le s - 1$, to the output gate of the circuit C_2 (thus, the variables $z_{k,j}$, $2^{|B_k|} \le j \le s-1$, are forced to have the value 0 in any satisfying assignment to C_2).

An enforcement circuitry is added to C_2 to ensure that every satisfying assignment to C_2 assigns the value 1 to at most one variable in each block B'_i. This can be achieved as follows. For every two distinct negative literals $\overline{z}_{i,j}$ and $\overline{z}_{i,h}$ in B'_i, $0 \le j, h \le 2^{|B_i|} - 1$, add an OR gate $g_{j,h}$. Connect $\overline{z}_{i,j}$ and $\overline{z}_{i,h}$ to $g_{i,h}$ and connect $g_{i,h}$ to the output AND gate of C_2. This completes the construction of the circuit C_2. The circuit C_2 is an antimonotone Π_t-circuit (again the enforcement circuitry does not increase the depth of C_2). Thus, (C_2, k) is an instance of the problem WCS$^-[t]$.

We verify that the circuit C_1 is satisfiable if and only if the circuit C_2 has a satisfying assignment of weight k. Suppose that the circuit C_1 is satisfied by an assignment τ. Let τ_i be the restriction of τ to block B_i, $1 \le i \le k$. Let j_i be the integer such that $\text{bin}_i(j_i) = \tau_i$. Consider the weight-$k$ assignment τ' to C_2 that for each i assigns $z_{i,j_i} = 1$ and all other variables in B'_i to 0. We show that τ' satisfies the circuit C_2. Let g_1 be a level-1 AND gate in C_1 that is satisfied by the assignment τ. Since C_2 is antimonotone, all inputs to g_1 in C_2 are negative literals. Since all negative literals except $\overline{z}_{i,j_i}$ in block B'_i have the value 1, we only have to prove that no $\overline{z}_{i,j_i}$ from any block B'_i is an input to g_1. Assume to the contrary that $\overline{z}_{i,j_i}$ in block B'_i is an input to g_1. Then by the construction of the circuit C_2, there is a variable x_q that is the h-th variable in block B_i such that either x_q is an input to g_1 in C_1 and the h-th bit of $\text{bin}_i(j_i)$ is 0, or $\overline{x}_q$ is an input to g_1 in C_1 and the h-th bit of $\text{bin}_i(j_i)$ is 1. However, by our construction of the index j_i from the assignment τ, if the h-th bit of $\text{bin}_i(j_i)$ is 0 then τ assigns $x_q = 0$, and if the h-th bit of $\text{bin}_i(j_i)$ is 1 then τ assigns $x_q = 1$. In either case, τ would not satisfy the gate g_1, contradicting our assumption. Thus, for all i, no $\overline{z}_{i,j_i}$ is an input to the gate g_1, and the assignment τ' satisfies the gate g_1. Since g_1 is an arbitrary level-1 AND gate in C_2, we conclude that the assignment τ' satisfies all level-1 AND gates in C_2 whose corresponding gates in C_1 are satisfied by the assignment τ. Since τ satisfies the circuit C_1, the weight-k assignment τ' satisfies the circuit C_2.

Conversely, suppose that the circuit C_2 is satisfied by a weight-k assignment τ'. Because of the enforcement circuitry in C_2, the assignment τ' assigns the value 1 to exactly one variable in each block B'_i (in particular, this variable in block B'_k must be one of the first $2^{|B_k|}$ variables in B'_k since the last $s - 2^{|B_k|}$ variables in B'_k are forced to have the value 0 in the satisfying assignment τ'). Suppose that in block B'_i, τ' assigns the value 1 to the variable z_{i,j_i}. Then we set an assignment $\tau_i = \text{bin}_i(j_i)$ to block B_i in C_1. Let τ be the assignment whose restriction on block B_i is τ_i. We prove that τ satisfies the circuit C_1. In effect, if a level-1 AND gate g_2 in C_2 is satisfied by the assignment τ', then no negative literal $\overline{z}_{i,j_i}$ is an input to g_2. Suppose that g_2 is not satisfied by τ in C_1, then either a positive literal x_q is an input to g_2 and τ assigns $x_q = 0$, or a negative literal $\overline{x}_q$ is an input to g_2 and τ assigns $x_q = 1$. Let x_q be the h-th variable in block B_i. If τ assigns $x_q = 0$ then the h-th bit in $\text{bin}_i(j_i)$ is 0. Thus, x_q cannot be an input to g_2 in C_1 because otherwise by our construction the negative literal $\overline{z}_{i,j_i}$ would be an input to g_2 in C_2. On the other hand, if τ assigns $x_q = 1$ then the h-th bit in $\text{bin}_i(j_i)$ is 1, thus, $\overline{x}_q$ cannot be an input to g_2 in C_1 because otherwise the negative literal $\overline{z}_{i,j_i}$ would be an input to g_2 in C_2. This contradiction shows that the gate g_2 must be satisfied by the assignment τ. Since g_2 is an arbitrary level-1 AND gate in C_2, we conclude that the assignment τ satisfies all level-1 AND gates in C_1 whose corresponding level-1 AND gates in C_2 are satisfied by the assignment τ'. Since τ' satisfies the circuit C_2, the assignment τ satisfies the circuit C_1 and hence the circuit C_1 is satisfiable.

This completes the proof that when t is odd, the Π_t-circuit C_1 is satisfiable if and only if the pair (C_2, k) is a yes-instance of WCS$^-[t]$.

Summarizing the above discussion, we conclude that for any $t \geq 2$, from a Π_t-circuit C_1 of n_1 input variables and size m_1, we can construct an instance (C_2, k) of the problem WCS$^*[t]$ such that C_1 is satisfiable if and only if (C_2, k) is a yes-instance of WCS$^*[t]$. Here $k = \lceil n_1/r \rceil$, and C_2 has $n_2 = 2^r k$ input variables and size $m_2 \leq m_1 + n_2 + k + k2^{2r} \leq 2m_1 + k2^{2r+1}$ (where the term $k + k2^{2r}$ is an upper bound on the size of the enforcement circuitry). Finally, it is straightforward to verify that the pair (C_2, k) can be constructed from the circuit C_1 in time $O(m_2^2)$.

Theorem 6. *Let $t \geq 2$ be an integer. For any function f, if the problem* WCS$^*[t]$ *is solvable in time $f(k)n^{o(k)}m^{O(1)}$, then the problem* SAT$[t]$ *can be solved in time $2^{o(n)}m^{O(1)}$.*

Proof. Suppose that there is an algorithm M_{WCS} of running time bounded by $f(k)n^{k/\lambda(k)}p(m)$ that solves the problem WCS$^*[t]$, where $\lambda(k)$ is a nondecreasing and unbounded function and p is a polynomial. Without loss of generality, we can assume that the function f is nondecreasing, unbounded, and that $f(k) \geq 2^k$. Define f^{-1} by $f^{-1}(h) = \max\{q \mid f(q) \leq h\}$. Since the function f is nondecreasing and unbounded, the function f^{-1} is also nondecreasing and unbounded, and satisfies $f(f^{-1}(h)) \leq h$. From $f(k) \geq 2^k$, we have $f^{-1}(h) \leq \log h$.

Now we solve the problem SAT$[t]$ as follows. For an instance C_1 of SAT$[t]$, where C_1 is a Π_t-circuit of n_1 input variables and size m_1, we set the integer $r = \lfloor 3n_1/f^{-1}(n_1) \rfloor$, and call the algorithm A_1 in Lemma 2 to convert C_1 into an instance (C_2, k) of the problem WCS$^*[t]$. Here $k = \lceil n_1/r \rceil$, C_2 is a Π_t-circuit of $n_2 = 2^r k$ input variables and size $m_2 \leq 2m_1 + 2^{2r+1}k$, and the algorithm A_1 takes time $O(m_2^2)$. According to Lemma 2, we can determine if C_1 is a yes-instance of SAT$[t]$ by calling the algorithm M_{WCS} to determine if (C_2, k) is a yes-instance of WCS$^*[t]$. The running time of the algorithm M_{WCS} on (C_2, k) is bounded by $f(k)n_2^{k/\lambda(k)}p(m_2)$. Combining all above we get an algorithm M_{sat} of running time $f(k)n_2^{k/\lambda(k)}p(m_2) + O(m_2^2)$ for the problem SAT$[t]$. We analyze the running time of the algorithm M_{sat} in terms of the values n_1 and m_1.

Since $k = \lceil n_1/r \rceil \leq f^{-1}(n_1) \leq \log n_1$,[2] we have $f(k) \leq f(f^{-1}(n_1)) \leq n_1$. Moreover,

$$k = \lceil n_1/r \rceil \geq n_1/r \geq n_1/(3n_1/f^{-1}(n_1)) = f^{-1}(n_1)/3.$$

Therefore if we set $\lambda'(n_1) = \lambda(f^{-1}(n_1)/3)$, then $\lambda(k) \geq \lambda'(n_1)$. Since both λ and f^{-1} are nondecreasing and unbounded, $\lambda'(n_1)$ is a nondecreasing and unbounded function of n_1. We have (note that $k \leq f^{-1}(n_1) \leq \log n_1$),

$$\begin{aligned} n_2^{k/\lambda(k)} &= (k2^r)^{k/\lambda(k)} \leq k^k 2^{kr/\lambda(k)} \leq k^k 2^{3kn_1/(\lambda(k) f^{-1}(n_1))} \leq k^k 2^{3n_1/\lambda(k)} \\ &\leq k^k 2^{3n_1/\lambda'(n_1)} = 2^{o(n_1)}. \end{aligned}$$

Finally, consider the factor m_2. Since f^{-1} is nondecreasing and unbounded,

$$m_2 \leq 2m_1 + k2^{2r+1} \leq 2m_1 + 2\log n_1 2^{6n_1/f^{-1}(n_1)} = 2^{o(n_1)} m_1.$$

Therefore, both terms $p(m_2)$ and $O(m_2^2)$ in the running time of the algorithm M_{sat} are bounded by $2^{o(n_1)} p'(m_1)$ for a polynomial p'. Combining all these, we conclude that the running time $f(k) n_2^{k/\lambda(k)} p(m_2) + O(m_2^2)$ of M_{sat} is bounded by $2^{o(n_1)} p'(m_1)$ for a polynomial p'. Hence, the problem SAT[t] can be solved in time $2^{o(n)} m^{O(1)}$. This completes the proof of the theorem.

The following corollary follows directly from the above theorem, and can be seen as as a generalization of Theorem 4 to higher levels of the W-hierarchy and the satisfiability problem. This result is due to Abrahamson et al. [1], and to Downey and Fellows [24]:

Corollary 1. *Let $t \geq 2$ be an integer. If $W[t] = FPT$ then the problem* SAT[t] *can be solved in time $2^{o(n)} m^{O(1)}$.*

In fact, Theorem 6 remains valid even if we restrict the parameter values to be bounded by an arbitrarily small function, as shown in the following theorem, whose proof is omitted and can be found in [14]:

Theorem 7. *Let $t \geq 2$ be an integer, and $\mu(n)$ a nondecreasing and unbounded function. If for a function f, the problem* WCS*[t] *is solvable in time $f(k) n^{o(k)} m^{O(1)}$ for parameter values $k \leq \mu(n)$, then the problem* SAT[t] *can be solved in time $2^{o(n)} m^{O(1)}$.*

The following theorem can be viewed as a generalization of Theorem 1:

Theorem 8. *For any $t \geq 2$, if* SAT[t] *can be solved in time $2^{o(n)} h(m)$ for some polynomial h, then $W[t-1] = FPT$.*

[2] Without loss of generality, we assume that in our discussions, all values under the ceiling function "$\lceil \cdot \rceil$" and the floor function "$\lfloor \cdot \rfloor$" are greater than or equal to 1. Therefore, we will always assume the inequalities $\lceil \beta \rceil \leq 2\beta$ and $\lfloor \beta \rfloor \geq \beta/2$ for any value β.

Proof. If $t = 2$, the theorem states that if CNF-SAT can be solved in time $2^{o(n)}h(m)$ then $W[1] = FPT$. This is basically the result in Theorem 1, which was proved in the previous section. Thus, we can assume that $t \geq 3$. Suppose that SAT$[t]$ is solvable in time $2^{o(n)}h(m)$. Then there exists an unbounded non-decreasing function $s(n)$ such that SAT$[t]$ can be solved in time bounded by $2^{n/s(n)}h(m)$. We distinguish two cases based on the parity of t.

Case 1. t is odd. We consider the WCS$^+[t-1]$ problem. Since this problem is complete for $W[t-1]$, it suffices to show that this problem can be solved in time $f(k)h'(m)$ where f is a function independent of the circuit size m, and h' is a polynomial. Let (C, k) be an instance of WCS$^+[t-1]$, where C has n input variables and size m. Since $t-1$ is even, the gates at level 1 in C are OR gates. Let $x_1, \dots, x_n$ be the input variables to C. We will construct a circuit C' from C with $k\lceil \log n \rceil$ input variables, such that C has a weight k assignment if and only if C' is satisfiable. The input variables in C' are divided into k blocks $B_1, \dots, B_k$, where block B_i, $i = 1, \dots, k$, consists of $r = \lceil \log n \rceil$ input variables $z_i^1, \dots, z_i^r$. Also, for every input variable z_i^j, $i \in \{1, \dots, k\}$, $j \in \{1, \dots, r\}$, we associate the input literal $\overline{z}_i^j$ to denote its negation. Informally speaking, each block B_i will contain the encoding of an input variable whose value is 1 in a weight-k assignment to C. We show how to connect the new input variables and their negations to the level-1 OR gates in C. Let g be a level-1 OR gate in C. Let x_p be an input to g, and let $b_1 b_2 \dots b_r$ be the binary representation of the number p (if there are fewer than r bits in the binary representation of p, we pad the binary representation of p with the appropriate number of 0's on the left to make it consist of exactly r bits). We introduce k new AND gates $g_p^1, \dots, g_p^k$. Each gate g_p^i, $i = 1, \dots, k$, has exactly r inputs, and its input comes only from input variables in block B_i and their negations. Informally speaking, each gate g_p^i will be satisfied if and only if block B_i contains the binary representation of p, and hence, encodes x_p. The input to gate g_p^i is determined as follows. For $j = 1, \dots, r$, if $b_j = 0$, then connect $\overline{z}_i^j$ to g_p^i, and if $b_j = 1$, then connect z_i^j to g_p^i. Now replace the connection from x_p to g by the connections from all gates $g_p^i, i = 1, \dots, k$, to g. We repeat this process for every level-1 gate g in C and every input variable in $\{x_1, \dots, x_n\}$ to g. Clearly, this construction only adds a single level to the circuit C consisting of AND gates, and hence, the resulting circuit is a Π_t circuit. We also add enforcement circuitry to ensure that the k blocks B_i, $i = 1, \dots, k$, encode distinct k variables. This can be simply achieved by adding a circuitry that performs a bitwise XOR operation to the corresponding variables in every two blocks, which can be done by adding a 3-level AND-of-OR-of-AND subcircuits to every two blocks (note that the last AND can be merged with the output AND gate of the circuit if $t = 3$). Clearly, the resulting circuit is still a Π_t-circuit. Moreover, the size of C is only increased by a polynomial factor in its original size. Let C'_F be the circuit resulting from this construction. From the above discussion we know that C' is a Π_t-circuit of size $h'(m)$ for some polynomial h'. Since the k input blocks in C' basically encode the k input variables in C with value 1 in a weight-k assignment to C, it is not difficult to verify that C has a weight-k truth assignment if and only if C' is satisfiable. Now C' is an instance of $\textsc{sat}[t]$ with kr input variables. It follows that we can decide if C' is satisfiable in time bounded by $T(n) = 2^{kr/s(kr)}h(h'(m)) = 2^{k\lceil \log n \rceil / s(k \lceil \log n \rceil)}h(h'(m)) \leq 2^{k(\log n + 1)/s'(n)}h''(m)$, for some unbounded non-decreasing function $s'(n)$, and some polynomial h''. Thus $T(n) \in 2^{o(\log n)k}h''(m)$, and $\textsc{wcs}^+[t-1]$ is solvable in time $2^{o(\log n)k}h''(m)$ for some polynomial h''. It follows that $\textsc{wcs}^+[t-1]$ is fixed parameter tractable (see Lemma 1), and hence, $W[t-1] = FPT$.

Case 2. t is even, and hence $t-1 \geq 3$ is odd. We consider the $\textsc{wcs}^-[t-1]$ problem, which is complete for $W[t-1]$. The proof proceeds in a very similar fashion to the proof of **Case 1** above. Let (C, k) be an instance of $\textsc{wcs}^-[t-1]$, and note that the gates at level 1 in C are AND gates. Let $\overline{x}_1, \ldots, \overline{x}_n$ be the input literals to C, and let r and B_i, $i = 1, \ldots, k$, be as defined above. Again, block B_i will be used to encode the indices of the input variables in C that are set to 1 in a weight-k assignment to C. Let g be a gate at level-1 in C, and suppose that $\overline{x}_p$, where $p \in \{1, \ldots, n\}$, is connected to g. Now $\overline{x}_p$ is 1 if and only if x_p is 0, if and only if none of the blocks B_i, $i = 1, \ldots, k$ contains the binary representation of p. Thus, in C' we will connect the new input variables to g as follows. We introduce k new OR gates $g_p^1, \ldots, g_p^k$. Each gate g_p^i, $i = 1, \ldots, k$, has exactly r inputs, and its input comes only from input variables in block B_i and their negations. Informally speaking, each gate g_p^i will be satisfied if and only if block B_i does not contain the binary representation of p, and hence, does not encode x_p. Suppose the binary representation of p is $b_1 b_2 \ldots b_r$. For $i = 1, \ldots, k$, the input to g_p^i is determined as follows. For $j = 1, \ldots, r$, if $b_j = 0$, then connect z_i^j to g_p^i, and if $b_j = 1$, then connect $\overline{z}_i^j$ to g_p^i. Now replace the connection from $\overline{x}_p$ to g by the connections from all gates g_p^i, $i = 1, \ldots, k$ to g, and repeat that for every level-1 gate in C and every original input literal to that gate. This adds an OR-level to C, thus increasing the number of levels in C by 1, and resulting in a Π_t-circuit. Now we can add the enforcement circuitry to ensure that all k blocks encode k distinct input variables. This can be simply achieved by adding a circuitry that performs a bitwise XOR operation to the corresponding variables in every two blocks. The resulting circuitry that tests that no two blocks are the same can be implemented by an OR-of-AND-of-AND-of-OR subcircuit (the last AND gate can be identified with the output gate of C if $t = 4$). Since $t \geq 4$, the resulting circuit C' is a Π_t-circuit whose size is not more than a polynomial in the size of C. The proof from this point on proceeds in exactly the same fashion as in **Case 1** above.

It follows that $W[t-1] = FPT$. This completes the proof.

5 The Miniaturization Classes

As we have seen in the previous section, the subexponential-time computability of the satisfiability problem on circuits of depth t, for $t \geq 2$, implies the fixed-parameter tractability of the class $W[t-1]$, and is implied by the fixed-parameter tractability of the class $W[t]$. Also, the failure of the ETH hypothesis is implied by the fixed-parameter tractability of the class $W[1]$. One may naturally ask whether the subexponential-time computability of these satisfiability problems, and also of other important NP-hard problems, is *equivalent* to the fixed-parameter tractability of some parameterized problems. In particular, is the hypothesis ETH *equivalent* to the fixed-parameter intractability of a particular parameterized problem?

The problem was initially considered in [9]. Downey *et al.* [23] formally proposed a process, named *parameterized miniaturization*, that establishes the equivalence between the subexponential-time computability of problems and the fixed-parameter tractability of their corresponding miniaturized parameterized problems. To describe this process, we need to be more careful in the use of the "size" of problem instances. Note that this had not been a problem for polynomial-time computation because any reasonable choice of instance size is polynomially related to the length of a reasonable encoding of the instance, and polynomial-time computation is robust for these variations. On the other hand, when we study subexponential-time computation, we implicitly allow only linear changes in the metric based on which the complexity of the computation is measured. Therefore, we have to be very careful in the use of certain conventional metrics for instance size, such as the number of Boolean variables in a satisfiability problem and the number of vertices in a graph. For example, if we use the number n of variables as a metric in the CNF-SAT problem, then CNF-SAT is certainly not solvable in subexponential-time in terms of n (i.e., in time $2^{o(n)}$) because the length of any encoding of an instance can be $2^{\Omega(n)}$. On the other hand, if we use the length l of a binary encoding of an instance of n variables and m clauses of CNF-SAT as the metric, then, because $l \geq (n+m)\log n$, CNF-SAT can be obviously solved in subexponential-time in l: a simple brute-force algorithm takes time $O(2^n nm \log n) = 2^{o(l)}$.

In the following discussion, we shall assume that we use a "natural" size for the problem instances. In particular, the size of a circuit will be the number of input variables plus the number of links in the circuit, the size of a Boolean formula will be the number of occurrences of literals in the formula, and the size of a graph is the number of its vertices plus the number of its edges.

We first consider the 3-SAT problem in terms of the size s of the input formula. The miniaturization process of the 3-SAT problem gives the following parameterized problem:

> MINI(3-SAT):
> Given nonnegative integers m and k in unary, and an instance F of size bounded by $k \log m$ for 3-SAT, where k is the parameter, decide if F is satisfiable.

The following theorem follows from a similar proof by Downey et al. [23] for circuit satisfiability:

Theorem 9 ([23]). *The parameterized problem* MINI(3-SAT) *is fixed parameter tractable if and only if the* 3-SAT *problem is solvable in subexponential time, i.e., in time $2^{o(s)}$, where s is the formula size.*

Proof. Suppose that 3-SAT is solvable in time $2^{o(s)}$ by an algorithm $\mathcal{A}$. Given an instance (F, m, k) of MINI(3-SAT), where the Boolean formula F has size bounded by $s = k \log m$, we simply invoke the algorithm $\mathcal{A}$ on the formula F to decide, in time $2^{o(s)} = 2^{o(k \log m)}$, whether F is satisfiable or not. By Lemma 1, this shows that MINI(3-SAT) is fixed-parameter tractable.

Conversely, suppose that MINI(3-SAT) is fixed parameter tractable, and hence is solvable by an algorithm $\mathcal{A}'$ in time $f(k)|x|^{O(1)}$, where f is a computable function of k, and $|x| = O(km)$ is the length of (any reasonable encoding of) the instance $x = (F, m, k)$ of MINI(3-SAT). Without loss of generality, we can assume that $f(k) \geq k$ for all k and that f is a strictly increasing function, from which we derive that the inverse function f^{-1} is well-defined and unbounded, and satisfies the condition $f^{-1}(n) \leq n$. For an instance F' of size s for the 3-SAT problem, let $k = f^{-1}(s)$ and $m = 2^{s/k}$, and consider the instance $x = (F', m, k)$ of MINI(3-SAT) (note that the size of F' is $s = k \log m$, and that $|x| = O(m + k + s \log s) = O(2^{s/f^{-1}(s)} s^2)$. By invoking the algorithm $\mathcal{A}'$ on (F', m, k), we can decide whether F' is satisfiable or not in time

$$f(k)|x|^{O(1)} = s^{O(1)} 2^{O(s/f^{-1}(s))} = 2^{o(s)},$$

where the last equality holds true because f^{-1} is a non-decreasing and unbounded function. This proves that 3-SAT can be solved in time $2^{o(s)}$, and hence, completes the proof of the theorem.

The 3-SAT problem measured by formula size is complete for the class SNP under serf-reductions, in the sense that if 3-SAT is solvable in subexponential time, then all problems in SNP are solvable in exponential time [30]. As mentioned before, there is a large number of important NP-hard problems that are complete for the class SNP under the serf-reduction, including the 3-SAT problem measured by *the number of input variables*. Therefore, the ETH hypothesis is equivalent to the statement that the 3-SAT problem measured by formula size is not subexponential-time solvable. This, combined with Theorem 9, gives us:

Theorem 10 ([23]). *The hypothesis ETH fails if and only if the parameterized problem* MINI(3-SAT) *is fixed-parameter tractable.*

The parameterized miniaturization process on SNP-complete problems under the serf-reduction enables the discovery of a class of parameterized problems whose fixed parameterized tractability is equivalent to the failure of the ETH hypothesis, for which the MINI(3-SAT) problem is a typical representative. Based on this observation, Downey et al. [23] introduced a new parameterized class $M[1]$ that consists of all parameterized problems that are fpt-reducible to the MINI(3-SAT) problem. In particular, the fixed parameter tractability of any $M[1]$-complete problem (under the fpt-reduction) is equivalent to the failure of the ETH hypothesis. The following are some examples of $M[1]$-complete problems, obtained based on the parameterized miniaturization process on "size-constrained" SNP-complete problems under serf-reductions [30]:

MINI(CIRC-SAT):
Given nonnegative integers k and m in unary, and a circuit C of size bounded by $k \log m$, where k is the parameter, decide if C is satisfiable.
MINI(IS):
Given nonnegative integers k and m in unary, a graph G of size bounded by $k \log m$, and a parameter r, decide if G have an independent set of at least r vertices.
MINI(VC):
Given nonnegative integers k and m in unary, a graph G of size bounded by $k \log m$, and a parameter r, decide if G have a vertex cover of at most r vertices.

The class $M[1]$ has the following relationships with the existing parameterized classes:

Theorem 11 ([23]). *$FPT \subseteq M[1] \subseteq W[1]$.*

Although the work of Cai and Juedes [9] implicitly hinted at the following result without reference to the class $M[1]$, the result was explicitly stated in Downey et al. [23], and follows from Theorem 10:

Theorem 12 ([23]). *$FPT = M[1]$ if and only if 3-SAT is solvable in time $2^{o(n)}$, where n is the number of variables in the input formula, and if and only if the hypothesis ETH fails.*

Theorem 12 provides a nice and precise characterization of the subexponential-time computability of many SNP-complete problems (under the serf-reduction) in terms of the fixed-parameter tractability of their corresponding miniaturized parameterized problems. However, there are still other important satisfiability problems, whose subexponential-time computability cannot be characterized by the theorem. Observing this, Chen, Flum, and Grohe have further considered the parameterized miniaturization process and derived the equivalence between the subexponential-time computability of problems and the fixed-parameter tractability of their corresponding miniaturized parameterized problems, for higher levels of the W-hierarchy [19–21, 25]. To describe this extension, we have to be further more careful with the notation of the instance size on which the exponential part in the computational complexity of a problem Q is measured. Typically, the *search-size* $\nu(x)$ of an instance x of Q is referred to the cardinality of a universal set U of which the instance x seeks a subset as its solution, and the *length* $|x|$ of x is the length of any reasonable encoding of the instance x. Thus, the computational complexity of the problem Q is measured by a function of the two metrics $\nu(x)$ and $|x|$. Note that in the above case, a simple enumeration algorithm of the subsets of the universal set U solves the problem in time $2^{\nu(x)}|x|^{O(1)}$. We say that the problem Q is *solvable in subexponential-time* if it can be solved in time $2^{o(\nu(x))}|x|^{O(1)}$.

For two integers $t \geq 1$ and $d \geq 1$, let us call a circuit C a *$\Pi_{t,d}$-circuit* if C is a Π_{t+1}-circuit in which the fan-in of each gate in level-1 is bounded by the integer constant d. Consider the satisfiability problem $\Pi_{t,d}$-SAT, which for a given $\Pi_{t,d}$-circuit C, asks if C is satisfiable. Chen and Grohe [21] introduced the following miniaturized problems:

> MINI($\Pi_{t,d}$-SAT):
> Given an instance C of $\Pi_{t,d}$-SAT, with a parameter $k = \lceil n/\log m \rceil$, where m is the size of C and n is the number of variables in C, decide if C is satisfiable.

We have the following theorem:

Theorem 13 ([21]). *For all $t \geq 1$, the satisfiability problem $\Pi_{t,d}$-SAT is subexponential-time solvable if and only if the parameterized problem* MINI($\Pi_{t,d}$-SAT) *is fixed-parameter tractable.*

Similar to the definition of the class $M[1]$, we can define new miniaturized classes of parameterized problems based on the fpt-reduction:

Definition 1. *For each integer $t \geq 2$, let $M[t]$ be the class of all parameterized problems that are fpt-reducible to the problem* MINI($\Pi_{t,d}$-SAT) *for some constant $d \geq 1$.*

Similar to Theorem 11, we obtain:

Theorem 14 ([21]). *For all $t \geq 2$, we have $W[t-1] \subseteq M[t] \subseteq W[t]$.*

The hierarchy $\bigcup_{t \geq 1} M[t]$ is called the *M-hierarchy*, which, by Theorem 14, refines the W-hierarchy:

$$FPT \subseteq M[1] \subseteq W[1] \subseteq \cdots \subseteq M[t-1] \subseteq W[t-1] \subseteq M[t] \subseteq W[t] \cdots$$

This study has also motivated the introduction of the following classification in nonparameterized problems that are solvable in exponential time.

Definition 2. *For $t \geq 1$, let $S[t]$ be the class consisting of all the problems that are serf-reducible to the problem $\Pi_{t,d}$-SAT, for some constant $d \geq 1$.*

For each $t \geq 1$, the parameterized class $M[t]$ is the image of the nonparameterized class $S[t]$ under the miniaturization mapping. The hierarchy $\bigcup_{t \geq 1} S[t]$ is called the *S-hierarchy.* The miniaturization process serves as a very nice mapping from nonparameterized problems solvable in exponential time to parameterized problems. More specifically, it maps an equivalence class $\mathcal{E}_1$ of exponential-time solvable problems (under the serf-reduction) to an equivalence class $\mathcal{E}_2$ of parameterized problems (under the fpt-reduction) such that $\mathcal{E}_1$ is subexponential-time solvable if and only if $\mathcal{E}_2$ is fixed parameter tractable. In particular, this mapping induces an "isomorphism" between the S-hierarchy, a nonparameterized complexity class, and the M-hierarchy, a parameterized complexity class [21].

6 Computational Lower Bounds

Theorem 5 shows that if the $W[1]$-complete problem WCNF 2-SAT$^-$ is solvable in time $f(k)m^{o(k)}$, then ETH fails. Theorem 6 states that for any $t \geq 2$, if the $W[t]$-complete problem WCS$^*[t]$ is solvable in time $f(k)n^{o(k)}m^{O(1)}$ then the satisfiability problem SAT$[t]$ can be solved in subexponential time. Note that the assumptions in these theorems are weaker than that of collapsing the W-hierarchy. On the other hand, they specify more detailed computational time bounds for the problems. Because of the hypothesis ETH, subexponential-time algorithms for 3-SAT and SAT$[t]$ for all $t \geq 2$ are unlikely. In this sense, Theorems 5 and 6 offer convincing lower bounds on the parameterized complexity for problems that are hard or complete for each level in the W-hierarchy. Interestingly enough, this line of research also implies lower bounds on computational complexity of approximation algorithms for several NP-hard optimization problems. In this section, we shall discuss recent developments in this line of research.

6.1 Lower Bounds on Parameterized Complexity

To discuss lower bounds for parameterized problems, we again need a more careful description of problem instances. For example, the WCS$^*[t]$ problem now has three different metrics for each of its instance (C, k): the length m of the instance (C, k), the parameter k, and the *search-size* n that is the number of input variables in C. We have seen from the previous section that, unless unlikely collapses occur in parameterized complexity theory, the problems WCS$^*[t]$ require computational time $f(k)n^{\Omega(k)}p(m)$, for any polynomial p and any function f. The dominating term in the time bound depends on the search-size n and the parameter k, instead of the instance length m.

Many well-known NP-hard problems have similar formulations. We list some of them here:

WEIGHTED CNF-SAT (abbreviated WCNF-SAT):
Given a CNF formula F, and an integer k, decide if there is an assignment of weight k that satisfies the formula F. Here the search-size is the number of input variables in F.

SET COVER:
Given a collection $\mathcal{F}$ of subsets in a universal set U, and an integer k, decide whether there is a subcollection of k subsets in $\mathcal{F}$ whose union is equal to U. Here the search-size is the cardinality of the collection $\mathcal{F}$.

HITTING SET:
Given a collection $\mathcal{F}$ of subsets in a universal set U, and an integer k, decide if there is a subset S of k elements in U such that S intersects every subset in $\mathcal{F}$. Here the search-size is the cardinality of the universal set U.

Many parameterized problems share the property that they seek a subset of k elements in a set of search-size n satisfying certain properties. In most of the problems that we consider, the search space can be easily identified. For example, for the problems INDEPENDENT SET and CLIQUE, the search space is the vertex set. Thus, each instance of a parameterized problems is associated with a triple (k, n, m), where k is the parameter, n is the *search-size* of the instance, and m

is the length (of any reasonable encoding) of the instance. We will call such an instance a (k, n, m)-*instance.*

Theorems 5 and 6 suggest that the $W[1]$-complete problem WCNF 2-SAT$^-$ and the $W[t]$-complete problem WCS$^*[t]$ for $t \geq 2$ seem to have very high parameterized complexity. In the following, we introduce a new reduction to identify problems in the corresponding classes that are at least as difficult as these problems.

Definition 3. *A parameterized problem Q is linearly fpt-reducible (shortly fpt$_l$-reducible) to a parameterized problem Q' if there exist a function f and an algorithm A such that on each (k, n, m)-instance x of Q, the algorithm A produces, in time $f(k)n^{o(k)}m^{O(1)}$, a (k', n', m')-instance x' of Q', where $k' = O(k)$, $n' = n^{O(1)}$, $m' = m^{O(1)}$, and x is a yes-instance of Q if and only if x' is a yes-instance of Q'.*

The fpt$_l$-reduction naturally introduces the hardness of parameterized problems.

Definition 4. *A parameterized problem Q_1 is $W[1]$-hard under the linear fpt-reduction, shortly $W_l[1]$-hard, if the problem* WCNF 2-SAT$^-$ *is fpt$_l$-reducible to Q_1. A parameterized problem Q_t is $W[t]$-hard under the linear fpt-reduction, shortly $W_l[t]$-hard, for $t \geq 2$ if the problem* WCS$^*[t]$ *is fpt$_l$-reducible to Q_t.*

Based on the above definitions and using Theorem 5 and Theorem 6, we immediately derive:

Theorem 15. *For $t \geq 2$, no $W_l[t]$-hard parameterized problem can be solved in time $f(k)n^{o(k)}m^{O(1)}$ for a function f, unless the problem* SAT$[t]$ *is solvable in time $2^{o(n)}m^{O(1)}$, which implies the collapsing $W[t-1] = FPT$.*

Theorem 16. *No $W_l[1]$-hard parameterized problem can be solved in time $f(k)m^{o(k)}$ for a function f, unless the ETH hypothesis fails, which is equivalent to the collapsing $M[1] = FPT$.*

In fact, many known fpt-reductions on parameterized problems proposed in the literature are fpt$_l$-reductions, or can be modified to become fpt$_l$-reductions. Using these fpt$_l$-reductions, we can immediately derive computational lower bounds for a large number of parameterized problems.

Theorem 17. *The following parameterized problems are $W_l[2]$-hard:* WCNF-SAT*,* SET COVER*,* HITTING SET*, and* DOMINATING SET*. Thus, unless the problem* SAT$[2]$ *is solvable in time $2^{o(n)}m^{O(1)}$, none of them can be solved in time $f(k)n^{o(k)}m^{O(1)}$ for any function f.*

To consider $W_l[1]$-hard problems, define WCNF h-SAT, where $h > 0$ is a fixed integer, to be the parameterized problem consisting of the pairs (F, k), where F is a CNF formula in which each clause is a disjunction of at most h literals and F has a satisfying assignment of weight k.

Theorem 18. *The following problems are $W_l[1]$-hard:* WCNF h-SAT *for any integer $h \geq 2$,* CLIQUE, *and* INDEPENDENT SET. *Thus, unless the ETH hypothesis fails, none of them can be solved in time $f(k)m^{o(k)}$ for any function f.*

Each of the problems in Theorem 17 and Theorem 18 can be solved by a trivial brute-force algorithm of running time $cn^k m^2$, where c is an absolute constant, which simply enumerates all possible subsets of k elements in the search space. A lot of research has sought new approaches to improve this trivial upper bound. One of the common approaches is to apply a more careful branch-and-bound search process trying to optimize the manipulation of local structures before each branch. Continuously improved algorithms for these problems have been developed based on improved local structure manipulations. It has even been proposed to automate the manipulation of local structures [40] in order to further improve the computational time.

Theorem 17 and Theorem 18, however, provide strong evidence that the power of this approach is quite limited in principle. The lower bound $f(k)n^{\Omega(k)}p(m)$ for the problems in Theorem 17 and the lower bound $f(k)m^{\Omega(k)}$ for the problems in Theorem 18, where f can be any function and p can be any polynomial, indicate that *no* local structure manipulation running in polynomial time or in time depending only on the target value k will obviate the need for exhaustive enumerations.

One might suspect that a particular parameter value (e.g., a very small parameter value or a very large parameter value) would help solving the problems in Theorem 17 and Theorem 18 more efficiently. This possibility is, unfortunately, denied by the following theorems, which indicate that, essentially, the problems are difficult for *every* parameter value.

Theorem 19. *For any constant ϵ, $0 < \epsilon < 1$, and for any nondecreasing and unbounded function μ satisfying $\mu(n) \leq n^\epsilon$, and $\mu(2n) \leq 2\mu(n)$, none of the problems in Theorem 17 can be solved in time $n^{o(k)}m^{O(1)}$ even if we restrict the parameter values k to $\mu(n)/8 \leq k \leq 16\mu(n)$, unless the problem* SAT[2] *is solvable in time $2^{o(n)}m^{O(1)}$, which implies $W[1] = FPT$.*

Note that the conditions on the function μ in Theorem 19 are satisfied by most complexity functions, such as $\mu(n) = \log\log n$ and $\mu(n) = n^{4/5}$. Therefore, for example, unless the problem SAT[2] is solvable in time $2^{o(n)}m^{O(1)}$, for any polynomial $p(m)$, constructing a hitting set of $\log\log n$ elements requires time $n^{\Omega(\log\log n)}p(m)$, and constructing a hitting set of $\sqrt{n}$ elements requires time $n^{\Omega(\sqrt{n})}p(m)$, where n is the size of the universal set U and m is the instance length.

Similar results hold for the problems in Theorem 18.

Theorem 20. *For any constant ϵ, $0 < \epsilon < 1$, and any nondecreasing and unbounded function μ satisfying $\mu(n) \leq n^\epsilon$, and $\mu(2n) \leq 2\mu(n)$, none of the problems in Theorem 18 can be solved in time $m^{o(k)}$ even if we restrict the parameter values k to $\mu(m)/8 \leq k \leq 16\mu(m)$, unless the ETH hypothesis fails.*

6.2 Refinements and Further Lower Bounds

The lower bounds on parameterized complexity in the previous subsection can be further strengthened based on more careful examinations of the relation between satisfiability problems and parameterized problems. Some of these strengthened results also require a stronger assumption on the complexity of satisfiability problems.

The efforts on achieving faster algorithms for satisfiability have been tremendous [37]. The current best algorithm for the CNF-SAT problem runs in time $2^{n(1-1/O(\log(m/n)))}m^{O(1)}$ [10]. Moreover, the current approaches do not seem to lead to break the time upper bound of the form $2^{n-o(n)}m^{O(1)}$ for solving the problem. In particular, designing an algorithm of running time $2^{\delta n}m^{O(1)}$, where $\delta < 1$ is a constant, seems to require a breakthrough. Impagliazzo and Paturi [30] conjectured that the CNF-SAT problem does not have an algorithm of running time $2^{\delta n}m^{O(1)}$, for a constant $\delta < 1$. Based on this conjecture, stronger computational lower bounds for parameterized problems can be achieved.

First, consider the DOMINATING SET problem: given a graph G of n vertices and a parameter k, decide if the graph G has a dominating set of at most k vertices. It is straightforward to solve the problem in time $O(n^{k+1})$ by enumerating every subset of at most k vertices in the graph and verifying if the subset makes a dominating set for the graph G. Based on fast matrix multiplication algorithms, we can slightly improve the above straightforward enumeration algorithm:

Proposition 1. ([39]) *The* DOMINATING SET *problem can be solved in time $n^{k+o(1)}$ for $k \geq 7$.*

One may suspect that by applying some algorithmic tricks, we may be able to further improve the algorithm for DOMINATING SET. Note that this was the case for the problem of finding a clique of size k, which can be solved in time $O(n^{(\omega/3)k}) = O(n^{0.793k})$, where $\omega < 2.376$ is the fastest matrix multiplication exponent. However, Patrascu and Williams have shown that such improvements would lead to a significant advancement in the research on CNF-SAT algorithms:

Theorem 21. ([39]) *For any constant $\epsilon > 0$, the* DOMINATING SET *problem cannot be solved in time $O(n^{k-\epsilon})$ unless the* CNF-SAT *problem can be solved in time $2^{\delta n}m^{O(1)}$ for some constant $\delta < 1$.*

Theorem 21 can be extended to other NP-hard problems based on effective reductions. For example, consider the SET COVER problem: given a collection $\mathcal{C}$ of n subsets of a universal set U of size m, decide if there are k subsets in $\mathcal{C}$ whose union is equal to U. Since the DOMINATING SET problem can be easily reduced to the SET COVER problem without changing the parameter value k, we derive directly that the SET COVER problem cannot be solved in time $O(n^{k-\epsilon})$ unless the CNF-SAT problem can be solved in time $2^{\delta n}m^{O(1)}$ for a constant $\delta < 1$.

These techniques have led to computational lower bounds for other interesting problems. The reader is referred to [39] for more results.

Recent research has further considered developing super-linear exponential-time lower bounds on parameterized problems.

A number of well-known parameterized problems went through the process of starting with super-linear exponential-time algorithms before advanced algorithmic techniques were developed that resulted in linear exponential-time algorithms for these problems. For example, the k-PATH problem (given a graph G and a parameter k, decide if the graph contains a simple path of length k) started with an algorithm of running time $2^k k! m^{O(1)}$, proposed in 1985 [35]. It was actually an open problem posted by Papadimitriou and Yannakakis whether k-PATH admits an algorithm running in time $2^{O(k)} m^{O(1)}$. Today, there is number of algorithms, based on at least three different new algorithmic techniques, for the k-PATH problem that run in time $c^k m^{O(1)}$, where c is a small constant [2, 4, 18, 41]. Other examples of this kind include the 3-D MATCHING and the 3-SET PACKING problems [13, 24, 32].

Therefore, the research in exponential-time algorithms could be still in a very premature stage, and one has to be very careful in conjecturing a super-linear exponential-time lower bound for a parameterized problem. On the other hand, very recent research shows that in certain cases, we can derive super-linear exponential-time lower bounds for parameterized problems based on certain beliefs about the complexity of satisfiability problems.

Consider the following CLOSEST STRING problem:

> CLOSEST STRING:
> Given a set of strings s_1, s_2, ..., s_t of the same length, and a parameter k, decide if there is a string s of the same length such that the Hamming distance between s and every s_i is bounded by k.

It has been known for a while [28] that the CLOSEST STRING problem can be solved in time $O(2^{k \log k} m)$, where m is the size of the input instance. A natural question is whether the exponential part, i.e., $2^{k \log k} = k^k$, in the complexity $O(2^{k \log k} m)$ can be improved to c^k for a constant c. Recent work by Lokshtanov, Marx, and Saurabh, shows that this is unlikely:

Theorem 22. ([33]) *Unless the hypothesis ETH fails, there is no $2^{o(k \log k)} m^{O(1)}$-time algorithm for the* CLOSEST STRING *problem.*

It was shown in [33] that for some other parameters the CLOSEST STRING problem also has super-linear exponential-time lower bounds. Other parameterized problems with super-linear exponential-time lower bounds can also be found in [33].

6.3 Lower Bounds on Approximation Algorithms

An interesting extension of the approach described in the previous subsections is deriving lower bounds on the computational complexity of approximation algorithms for NP-hard problems. We first give a brief review on the terminologies in approximation algorithms.

An *NP optimization problem* Q is a quadruple (I_Q, S_Q, f_Q, opt_Q), where

- I_Q is the set of input instances. It is recognizable in polynomial time;
- For each instance $x \in I_Q$, $S_Q(x)$ is the set of feasible solutions for x, which is defined by a polynomial p and a polynomial time computable predicate π (p and π only depend on Q) as $S_Q(x) = \{y : |y| \le p(|x|) \text{ and } \pi(x, y)\}$;
- $f_Q(x, y)$ is the objective function mapping a pair $x \in I_Q$ and $y \in S_Q(x)$ to a non-negative integer. The function f_Q is computable in polynomial time;
- $opt_Q \in \{\max, \min\}$. Q is called a *maximization problem* if $opt_Q = \max$, and a *minimization problem* if $opt_Q = \min$.

An *optimal solution* y_0 for an instance $x \in I_Q$ is a feasible solution in $S_Q(x)$ such that $f_Q(x, y_0) = opt_Q\{f_Q(x, z) \mid z \in S_Q(x)\}$. We will denote by $opt_Q(x)$ the value $opt_Q\{f_Q(x, z) \mid z \in S_Q(x)\}$.

An algorithm A is an *approximation algorithm* for an NP optimization problem Q if, for each input instance x in I_Q, the algorithm A returns a feasible solution $y_A(x)$ in $S_Q(x)$. The approximation algorithm A has an *approximation ratio* $r(m)$ if for any instance x in I_Q, the solution $y_A(x)$ constructed by the algorithm A satisfies the following condition:

- $opt_Q(x)/f_Q(x, y_A(x)) \le r(|x|)$ if Q is a maximization problem;
- $f_Q(x, y_A(x))/opt_Q(x) \le r(|x|)$ if Q is a minimization problem.

An NP optimization problem Q has a *polynomial-time approximation scheme* (PTAS) if there is an algorithm $A_Q(x, \epsilon)$ such that for each fixed real number $\epsilon_0 > 0$, $A_Q(x, \epsilon_0)$ is a polynomial-time approximation algorithm for the problem Q whose approximation ratio is bounded by $1 + \epsilon_0$.

The following "parameterization process" for NP optimization problems has been proposed in the literature.

Definition 5. *Let $Q = (I_Q, S_Q, f_Q, opt_Q)$ be an NP optimization problem. The parameterized version of Q is defined as follows:*

- *If Q is a maximization problem, then the parameterized version of Q is defined as $Q_{\ge} = \{(x, k) \mid x \in I_Q \text{ and } opt_Q(x) \ge k\}$;*
- *If Q is a minimization problem, then the parameterized version of Q is defined as $Q_{\le} = \{(x, k) \mid x \in I_Q \text{ and } opt_Q(x) \le k\}$.*

The above definition offers the possibility to study the relationship between the approximability and the parameterized complexity of NP optimization problems.

Theorem 23. *Let Q be an NP optimization problem. If the parameterized version of Q is $W_l[1]$-hard, then Q has no PTAS of running time $f(1/\epsilon)m^{o(1/\epsilon)}$ for any function f, unless the ETH hypothesis fails.*

Proof. We consider the case that $Q = (I_Q, S_Q, f_Q, opt_Q)$ is a maximization problem such that the parameterized version $Q_{\ge}$ of Q is $W_l[1]$-hard.

Suppose to the contrary that Q has a PTAS A_Q of running time $f(1/\epsilon)m^{o(1/\epsilon)}$ for a function f. We show how to use the algorithm A_Q to solve the parameterized problem $Q_{\ge}$. Consider the following algorithm $A_{\ge}$ for $Q_{\ge}$:

Algorithm $A_{\geq}$:
On an instance (x, k) of $Q_{\geq}$, call the PTAS algorithm A_Q on the instance x of Q with the real number $\epsilon = 1/(2k)$. Suppose that A_Q returns a solution y in $S_Q(x)$. If $f_Q(x, y) \geq k$, then return "yes", otherwise return "no".

We verify that the algorithm $A_{\geq}$ solves the parameterized problem $Q_{\geq}$. Since Q is a maximization problem, if $f_Q(x, y) \geq k$ then obviously $opt_Q(x) \geq k$. Thus, the algorithm $A_{\geq}$ returns a correct decision in this case. On the other hand, suppose $f_Q(x, y) < k$. Since $f_Q(x, y)$ is an integer, we have $f_Q(x, y) \leq k - 1$. Since A_Q is a PTAS for Q and $\epsilon = 1/(2k)$, we must have

$$opt_Q(x)/f_Q(x, y) \leq 1 + 1/(2k).$$

From this we get (note that $f_Q(x, y) < k$)

$$opt_Q(x) \leq f_Q(x, y) + f_Q(x, y)/(2k) \leq k - 1 + 1/2 = k - 1/2 < k.$$

Thus, in this case the algorithm $A_{\geq}$ also returns a correct decision. This proves that the algorithm $A_{\geq}$ solves the parameterized version $Q_{\geq}$ of the problem Q. The running time of the algorithm $A_{\geq}$ is dominated by that of the algorithm A_Q, which by our hypothesis is bounded by $f(1/\epsilon)m^{o(1/\epsilon)} = f(2k)m^{o(k)}$. Thus, the $W_l[1]$-hard problem $Q_{\geq}$ is solvable in time $f(2k)m^{o(k)}$. By Theorem 16, this implies that the ETH hypothesis fails.

The proof is similar for the case when Q is a minimization problem, and hence is omitted.

We demonstrate an application for Theorem 23. We pick the problem DISTINGUISHING SUBSTRING SELECTION as an example, which has drawn a lot of attention recently because of its applications in computational biology such as in drug generic design [22].

Consider all strings over a fixed alphabet. Denote by $|s|$ the length of the string s. The *distance* $D(s_1, s_2)$ between two strings s_1 and s_2, $|s_1| \leq |s_2|$, is defined as follows. If $|s_1| = |s_2|$, then $D(s_1, s_2)$ is the Hamming distance between s_1 and s_2; and if $|s_1| \leq |s_2|$, then $D(s_1, s_2)$ is the minimum of $D(s_1, s_2')$ over all substrings s_2' of length $|s_1|$ in s_2.

Based on the standard formulation of NP optimization problems, the (optimization version of the) DISTINGUISHING SUBSTRING SELECTION problem (DSSP) is defined as follows:

Definition 6. The DSSP problem is a quadruple (I_D, S_D, f_D, opt_D), where

- The instance set I_D is the set of tuples of the form (n, S_b, S_g, d_b, d_g), where n, d_b, and d_g are integers, $d_b \leq d_g$, $S_b = \{b_1, \ldots, b_{n_b}\}$ is a set of (bad) strings, $|b_i| \geq n$, and $S_g = \{g_1, \ldots, g_{n_g}\}$ is a set of (good) strings, $|g_j| = n$;
- For an instance $x = (n, S_b, S_g, d_b, d_g)$ in I_D, the solution set $S_D(x)$ consists of all strings of length n;

- For an instance $x = (n, S_b, S_g, d_b, d_g)$ in I_D and a solution $s \in S_D(x)$, the objective function value $f_D(x, s)$ is defined to be the largest non-negative integer d such that (i) $d \leq d_g$; (ii) $D(s, b_i) \leq d_b(2 - d/d_g)$ for all $b_i \in S_b$; and (iii) $D(s, g_j) \geq d$ for all $g_j \in S_g$. If such an integer d does not exist, then define $f_D(x, s) = 0$;
- $opt_D = \max$

Note that for $x \in I_D$ and $s \in S_D(x)$, the value $f_D(x, s)$ can be computed in polynomial time by checking each number $d = 0, 1, \ldots, d_g \leq n$.

Note that the objective of the DSSP problem is to find a string s that maximizes the value $f_D(x, s)$, which is bounded by d_g. In particular, if a string s can achieve $f_D(x, s) = d_g$, then s satisfies $D(s, b_i) \leq d_b$ for all $b_i \in S_b$ (i.e., the string s is similar enough to all bad strings) and $D(s, g_j) \geq d_g$ for all $g_j \in S_g$ (i.e., the string s is sufficiently different from all good strings).

The DSSP problem is NP-hard [27]. Deng et al. [22] developed a PTAS for DSSP whose running time is bounded by $O(m(n_b + n_g)^{O(1/\epsilon^6)})$, where m is the size of the instance.[3] Obviously, such an algorithm is not practical even for moderate values of the error bound ϵ. The question is, can we develop significantly faster PTAS for the DSSP problem?

Using the above parameterization process, we can parameterize the DSSP problem, and study the complexity of the corresponding parameterized problem DSSP$_{\geq}$.

Lemma 3. ([14]) *The parameterized problem* DSSP$_{\geq}$ *is* $W_l[1]$*-hard.*

From Lemma 3 and Theorem 23, we get the following result.

Theorem 24. *Unless the ETH hypothesis fails, the problem* DSSP *has no PTAS whose running time is bounded by* $f(1/\epsilon)m^{o(1/\epsilon)}$ *for any function* f.

Therefore, Theorem 24 implies that any PTAS for DSSP cannot run in time $f(1/\epsilon)m^{o(1/\epsilon)}$ for any function f. Thus essentially, no PTAS for DSSP can be practically efficient even for moderate values of the error bound ϵ. This is the first time a specific lower bound is derived on the running time of a PTAS for an NP-hard problem.

Lemma 3 is proved by a linear fpt-reduction from the DOMINATING SET problem to the problem DSSP$_{\geq}$, which leads to the computational lower bounds on PTAS for the DSSP problem in Theorem 24. This approach demonstrates an interesting property of this technique. In most cases, computational lower bounds and inapproximability of optimization problems are derived based on approximation ratio-preserving reductions [3], by which if a problem Q_1 is reduced to another problem Q_2, then Q_2 is at least as hard as Q_1. In particular, if

[3] In fact, the formulations of the optimization versions of the DSSP problem and its PTAS given in [22] look very different from the versions presented here. A proof is given in [14] that shows the equivalences of the problem formulation and PTAS given in [22] and that presented here.

Q_1 is reduced to Q_2 under an approximation ratio-preserving reduction, then the approximability of Q_2 is at least as difficult as that of Q_1. Therefore, the intractability of an "easier" problem in general cannot be derived using such a reduction from a "harder" problem. On the other hand, our computational lower bound on DSSP was obtained by a linear fpt-reduction from DOMINATING SET. It is well-known that DOMINATING SET has no polynomial time approximation algorithms of constant ratio [3], while DSSP has a PTAS. Thus, from the viewpoint of approximation, DOMINATING SET is much harder than DSSP, and our linear fpt-reduction reduces a harder problem to an easier problem. This hints that this approach for deriving computational lower bounds *cannot* be simply replaced by the standard approaches based on approximation ratio-preserving reductions.

Readers who are interested in the relation between fixed-parameter tractability and the efficiency of approximation algorithms for NP-hard optimization problems are referred to [8, 15, 34] for more discussions and details.

7 Concluding Remarks and Open Problems

The study of parameterized intractability and subexponential-time computability has significantly promoted new research directions in both complexity theory and algorithms. From the computational complexity viewpoint, this study has motivated the development of new frameworks whose intrinsic relations are being studied. From the algorithmics viewpoint, this study has motivated the invention of new algorithmic tools, beyond the world of polynomial-time computation, and has established connections among a large variety of computational problems. This demonstrates the robustness of computational intractability, thus providing convincing evidence of the existing computational lower bounds on problems of theoretical and practical importance, based on parameterized complexity and subexponential-time hypotheses.

The classification of parameterized intractability, i.e., the W-hierarchy, offers a framework for the study of computational intractability, which is a refinement of that of classical complexity theory. On the other hand, little is known about the structural properties of this hierarchy. For example, any natural "hierarchy collapsing" results about the W-hierarchy are still lacking. In particular, the following question remains unanswered: Does $FPT = W[t]$ $(t \geq 1)$ imply $FPT = W[s]$ for $s > t$? Note that because of the close connections between the W-hierarchy and the computational complexity of circuit satisfiability, the corresponding questions about circuit satisfiability problems are also important and significant: Would the subexponential-time computability of SAT$[t]$ imply the subexponential-time computability of SAT$[s]$ for $s > t$? In particular, would the failure of the ETH hypothesis imply the subexponential-time computability of CNF-SAT?

The current state of knowledge about the complexity of satisfiability problems seems to provide no hints on the above questions. For example, the 3-SAT problem can be solved in time $2^{\delta n} m^{O(1)}$, where $\delta < 0.56$ [5]. However, this does not seem to offer any ideas for solving the general CNF-SAT problem in time $2^{\delta' n}m^{O(1)}$ for any constant $\delta' < 1$. In fact, the current techniques used for solving 3-SAT do not even seem to be generalizable to CNF-SAT with bounded fan-in of level-1 gates (i.e., h-SAT for integer constants $h \geq 3$). It will be very interesting to investigate this direction. For example, is there a relation between the ETH hypothesis and the hypothesis "CNF-SAT is not solvable in time $2^{\delta n}m^{O(1)}$ for any constant $\delta < 1$"? Would a subexponential-time algorithm for 3-SAT imply a $2^{\delta n}m^{O(1)}$-time algorithm for CNF-SAT for some constant $\delta < 1$, or vice versa?

This reminds us of the well-known research line in complexity theory on the computational lower bounds for bounded-depth circuit computation, where it has been known that depth-t circuits are strictly more powerful than depth-s circuits for $t > s$ [29]. In fact, the fan-in of level-1 gates of a circuit is determinative of its computational power: for any $h \geq 2$, depth-h circuits in which the fan-in of level-1 gates is bounded by t is strictly more powerful than depth-h circuits in which the fan-in of level-1 gates is bounded by s, if $t > s$ [7]. Note that, these significant results are on the difference of computational power of circuit models. On the other hand, in the study of parameterized complexity and the complexity of satisfiability problems, the circuit depths in the problems are part of the "descriptive complexity" of the problems. Are there any correlations that can provide further insight in this direction? If we view a nondeterministic computation as a "guess-then-check" process [6], then the SAT[t] problems for all $t \geq 2$ require the same guessing power (i.e., picking a proper subset of the input variables), but differ strictly in the verification power (SAT[t] requires depth-t circuits for verification). The current research status in satisfiability algorithms seems to suggest that the difference in the verification power forces a difference in the deterministic computational complexity of the problems.

Another interesting research direction is on the algorithmics side. As stated in Theorem 21, a very sharp parameterized lower bound (such as $\Omega(n^k)$) for the DOMINATING SET problem seems to have consequences on exact algorithms for the important satisfiability problem CNF-SAT. This line of research has yielded a collection of results, formulated as "if problem A can be solved in time $t(\cdot)$ then problem B can be solved in time $s(\cdot)$," where $t(\cdot)$ and $s(\cdot)$ are precise functions (without hidden constants in their asymptotic notations). Is there a systematic method to relate such results? Such method will bear significant impact on the existence of more efficient exact algorithms for certain problems, and can be read from two angles. From the positive angle, those results suggest a way for improving the algorithms for a problem B under a certain framework (such as exact computability) by improving the algorithms for another problem A with respect to (possibly) a different computational framework (such as parameterized complexity). From the negative angle, such a result can serve as an indicator of the intricate difficulty of the computability of a problem A with respect to a certain computational framework, based on that of another problem B with respect to a different framework.

We also want to remark that the above discussions provide "observations" based on the current understanding and techniques in complexity theory and algorithmic research. For instance, the statement that the CNF-SAT problem cannot be solved in time $2^{\delta n} m^{O(1)}$ for a constant $\delta < 1$ seems to be very strong, and perhaps needs further investigation. One has to be more careful when using such statements as "hypotheses." Instead, these studies and observations should provide an impetus for new research insights, new ideas, and new techniques. All our algorithmic techniques for solving NP-hard problems, such as the satisfiability problems, are more or less based on enumerations. Thus, a more ambitious question would be to investigate new approaches for tackling NP-hard problems.

References

1. Abrahamson, K., Downey, R., Fellows, M.: 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. Alon, N., Yuster, R., Zwick, U.: Color-coding. Journal of the ACM 42, 844–856 (1995)
3. Ausiello, G., Protasi, M., Marchetti-Spaccamela, A., Gambosi, G., Crescenzi, P., Kann, V.: Complexity and Approximation: Combinatorial Optimization Problems and Their Approximability Properties, 1st edn. Springer-Verlag New York, Inc., Secaucus (1999)
4. Björklund, A.: Determinant sums for undirected hamiltonicity. In: Proceedings of the 51th Annual IEEE Symposium on Foundations of Computer Science, pp. 173–182 (2010)
5. Brüggemann, T., Kern, W.: An improved deterministic local search algorithm for 3SAT. Theoretical Computer Science 329, 303–313 (2004)
6. Cai, L., Chen, J.: On the amount of nondeterminism and the power of verifying. SIAM Journal on Computing 26(3), 733–750 (1997)
7. Cai, L., Chen, J., Håstad, J.: Circuit bottom fan-in and computational power. SIAM Journal on Computing 27(2), 341–355 (1998)
8. Cai, L., Fellows, M., Juedes, D., Rosamond, F.: The complexity of polynomial-time approximation. Theory of Computing Systems 41(3), 459–477 (2007)
9. Cai, L., Juedes, D.: On the existence of subexponential parameterized algorithms. Journal of Computer and System Sciences 67(4), 789–807 (2003)
10. Calabro, C., Impagliazzo, R., Paturi, R.: A duality between clause width and clause density for SAT. In: IEEE Conference on Computational Complexity, pp. 252–260 (2006)
11. Chen, J.: Characterizing parallel hierarchies by reducibilities. Information Processing Letters 39(6), 303–307 (1991)
12. Chen, J., Chor, B., Fellows, M., Huang, X., Juedes, D., Kanj, I., Xia, G.: Tight lower bounds for certain parameterized NP-hard problems. Information & Computation 201(2), 216–231 (2005)
13. Chen, J., Feng, Q., Liu, Y., Lu, S., Wang, J.: Improved deterministic algorithms for weighted matching and packing problems. Theoretical Computer Science 412(23), 2503–2512 (2011)
14. Chen, J., Huang, X., Kanj, I., Xia, G.: Strong computational lower bounds via parameterized complexity. Journal of Computer and System Sciences 72(8), 1346–1367 (2006)
15. Chen, J., Huang, X., Kanj, I., Xia, G.: Polynomial time approximation schemes and parameterized complexity. Discrete Applied Mathematics 155(2), 180–193 (2007)

16. Chen, J., Kanj, I., Jia, W.: Vertex cover: further observations and further improvements. Journal of Algorithms 41, 280–301 (2001)
17. Chen, J., Kanj, I., Xia, G.: On parameterized exponential time complexity. Theoretical Computer Science 410(27-29), 2641–2648 (2009)
18. Chen, J., Kneis, J., Lu, S., Molle, D., Richter, S., Rossmanith, P., Sze, S.-H., Zhang, F.: Randomized divide-and-conquer: improved path, matching, and packing algorithms. SIAM Journal on Computing 38, 2526–2547 (2009)
19. Chen, Y., Flum, J.: On miniaturized problems in parameterized complexity theory. Theoretical Computer Science 351(3), 314–336 (2006)
20. Chen, Y., Flum, J.: Subexponential time and fixed-parameter tractability: Exploiting the miniaturization mapping. Journal of Logic and Computation 19(1), 89–122 (2009)
21. Chen, Y., Grohe, M.: An isomorphism between subexponential and parameterized complexity theory. SIAM Journal on Computing 37(4), 1228–1258 (2007)
22. Deng, X., Li, G., Li, Z., Ma, B., Wang, L.: Genetic design of drugs without side-effects. SIAM Journal on Computing 32, 1073–1090 (2003)
23. Downey, R., Estivill-Castro, V., Fellows, M., Prieto-Rodriguez, E., Rosamond, F.: Cutting up is hard to do: the parameterized complexity of k-Cut and related problems. Electronic Notes in Theoretical Computer Science 78, 205–218 (2003)
24. Downey, R., Fellows, M.: Parameterized Complexity. Springer, New York (1999)
25. Flum, J., Grohe, M.: Parametrized complexity and subexponential time (column: Computational complexity). Bulletin of the EATCS 84, 71–100 (2004)
26. Flüm, J., Grohe, M.: Parameterized Complexity Theory. Springer, Berlin (2010)
27. Gramm, J., Guo, J., Niedermeier, R.: On Exact and Approximation Algorithms for Distinguishing Substring Selection. In: Lingas, A., Nilsson, B.J. (eds.) FCT 2003. LNCS, vol. 2751, pp. 195–209. Springer, Heidelberg (2003)
28. Gramm, J., Niedermeier, R., Rossmanith, P.: Fixed-parameter algorithms for closest string and related problems. Algorithmica 37(1), 25–42 (2003)
29. Håstad, J.: Computational limitations of small depth circuits. Technical report, MIT Press (1986)
30. Impagliazzo, R., Paturi, R.: On the complexity of k-SAT. Journal of Computer and System Sciences 62(2), 367–375 (2001)
31. Impagliazzo, R., Paturi, R., Zane, F.: Which problems have strongly exponential complexity? Journal of Computer and System Sciences 63(4), 512–530 (2001)
32. Koutis, I.: Faster Algebraic Algorithms for Path and Packing Problems. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part I. LNCS, vol. 5125, pp. 575–586. Springer, Heidelberg (2008)
33. Lokshtanov, D., Marx, D., Saurabh, S.: Slightly superexponential parameterized problems. In: Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 760–776 (2011)
34. Marx, D.: Parameterized complexity and approximation algorithms. The Computer Journal 51(1), 60–78 (2008)
35. Monien, B.: How to find long paths efficiently. Ann. Discrete Math. 25, 239–254 (1985)
36. Niedermeier, R.: Invitation to Fixed-Parameter Algorithms. Oxford University Press, USA (2006)
37. International Conference on Theory and Applications of Satisfiability Testing, `http://www.satisfiability.org/`
38. Papadimitriou, C., Yannakakis, M.: Optimization, approximation, and complexity classes. Journal of Computer and System Sciences 43, 425–440 (1991)

39. Patrascu, M., Williams, R.: On the possibility of faster SAT algorithms. In: Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 1065–1075 (2010)
40. Robson, J.M.: Algorithms for maximum independent sets. Journal of Algorithms 7, 425–440 (1986)
41. Williams, R.: Finding paths of length k in $O^*(2^k)$ time. Information Processing Letters 109(6), 315–318 (2009)
42. Williams, R.: Non-uniform ACC circuit lower bounds. In: IEEE Conference on Computational Complexity, pp. 115–125. IEEE Computer Society Press (2011)
43. Woeginger, G.: Exact Algorithms for NP-Hard Problems: A Survey. In: Jünger, M., Reinelt, G., Rinaldi, G. (eds.) Combinatorial Optimization (Edmonds Festschrift). LNCS, vol. 2570, pp. 185–207. Springer, Heidelberg (2003)