Tight Lower Bounds for Certain Parameterized NP-Hard Problems

Jianer Chen[*] Benny Chor[†] Mike Fellows[‡] Xiuzhen Huang[*]
David Juedes[¶] Iyad A. Kanj[‖] Ge Xia[*]

Abstract

Based on the framework of parameterized complexity theory, we derive tight lower bounds on the computational complexity for a number of well-known NP-hard problems. We start by proving a general result, namely that the parameterized weighted satisfiability problem on depth-t circuits cannot be solved in time $n^{o(k)}\text{poly}(m)$, where n is the circuit input length, m is the circuit size, and k is the parameter, unless the $(t-1)$-st level $W[t-1]$ of the W-hierarchy collapses to FPT. By refining this technique, we prove that a group of parameterized NP-hard problems, including WEIGHTED SAT, DOMINATING SET, HITTING SET, SET COVER, and FEATURE SET, cannot be solved in time $n^{o(k)}\text{poly}(m)$, where n is the size of the universal set from which the k elements are to be selected and m is the instance size, unless the first level $W[1]$ of the W-hierarchy collapses to FPT. We also prove that another group of parameterized problems which includes WEIGHTED q-SAT (for any fixed $q \geq 2$), CLIQUE, and INDEPENDENT SET, cannot be solved in time $n^{o(k)}$ unless all search problems in the syntactic class SNP, introduced by Papadimitriou and Yannakakis, are solvable in subexponential time. Note that all these parameterized problems have trivial algorithms of running time either $n^k\text{poly}(m)$ or $O(n^k)$.

1 Introduction

Solving well-known NP-hard problems with small parameter values has found important applications recently in areas such as computational biology. For instance, the Computational Biochemistry Research Group at the ETH Zürich has successfully applied the algorithms for the VERTEX COVER problem (determine whether a given graph G has a vertex cover of size k) to their research in multiple sequence alignments [15, 21], where the parameter value k can be bounded by 100. In the study of motif finding problem in computational biology, Pevzner and Sze [20] proposed a graph theoretical formulation that requires finding cliques of size k, where a typical value of k is 20. This approach has been followed by a steady stream of combinatorial approaches trying to improve the performance of motif finding algorithms (see, e.g., Buhler and Tompa [3] and their references).

However, from the parameterized complexity point of view [11], these two problems are very different. The VERTEX COVER problem is *fixed-parameter tractable* in the sense that it can be solved in time $f(k)n^c$, where $f(k)$ is a function of the parameter k and c is a fixed constant. After many rounds of improvement, the best known algorithm for VERTEX COVER runs in time $O(1.285^k + kn)$ [7], which has been implemented and is quite practical for parameter values up to 400 [5]. On

[*]Department of Computer Science, Texas A&M University, College Station, TX 77843, USA. This research is supported in part by the NSF grants CCR-0000206 and CCR-0311590. Email: {chen,xzhuang,gexia}@cs.tamu.edu.

[†]Department of Computer Science, Technion, Haifa 32000, Israel. Email: benny@cs.technion.ac.il.

[‡]School of Electrical Engineering and Computer Science, University of Newcastle, University Drive, Callaghan 23-8, Australia. Email: mfellows@cs.newcastle.edu.au.

[¶]School of EE&CS, Ohio University, Athens, OH 45701, USA. Email: juedes@ohiou.edu.

[‖]The corresponding author. School of CTI, DePaul University, 243 S. Wabash Avenue, Chicago, IL 60604, USA. Email: ikanj@cs.depaul.edu. This work was supported in part by DePaul University Competitive Research Grant.

the other hand, the CLIQUE problem is $W[1]$-hard, and thus, it is unlikely to be fixed-parameter tractable. The best known algorithm for finding a clique of size k in a graph of n vertices runs in time $O(n^{0.8k})$ [17], based on a combination of exhaustive search and the best matrix multiplication algorithm [8]. Obviously, such an algorithm is not practically feasible even for parameter values as small as 20.

On the surface, the $W[1]$-hardness of CLIQUE implies that any algorithm of running time $O(n^h)$ solving the problem *must* have the degree h of the polynomial n^h a function of the parameter k. However, this does not exclude the possibility that CLIQUE becomes feasible for small values of the parameter k. For instance, if CLIQUE is solvable by an algorithm running in time $O(n^{\lg \lg k})$, then such an algorithm is still feasible for moderately small values of k.[1]

In this paper we show that using the notion of W-hardness, we can prove much stronger lower bounds on the computational complexity for a large group of NP-hard parameterized problems, including the well-known problems WEIGHTED SAT, DOMINATING SET, HITTING SET, SET COVER, FEATURE SET, INDEPENDENT SET, and CLIQUE. These problems share a common property that each instance of size m of the problems has a universal set U of size n, and we are looking for a subset of k elements in U that meets certain given conditions. Note that all these problems have trivial algorithms of running time $n^k\text{poly}(m)$, which simply enumerate all subsets of size k in the universal set U to find a subset satisfying the given conditions in case it exists.

We start by developing a general result showing that in each level of the W-hierarchy, there is a natural parameterized problem that has time complexity $n^{\Omega(k)}\text{poly}(m)$ unless an unlikely collapse occurs in the parameterized complexity theory. More specifically, we prove that for every $t \geq 2$, the weighted satisfiability problem on depth-t circuits cannot be solved in time $n^{o(k)}\text{poly}(m)$, where n is the circuit input length and m is the circuit size, unless the $(t-1)$-st level $W[t-1]$ of the W-hierarchy collapses to FPT. By refining this technique and employing proper reductions, we are able to prove that, unless the first level $W[1]$ of the W-hierarchy collapses to FPT, a group of $W[2]$-hard parameterized problems cannot be solved in time $n^{o(k)}\text{poly}(m)$, where n is the size of the universal set from which the k elements are to be selected and m is the instance size. This group of $W[2]$-hard problems includes the well-known problems: WEIGHTED SAT, DOMINATING SET, HITTING SET, SET COVER, and FEATURE SET. Note that these results demonstrate that, under the assumption $W[1] \neq \text{FPT}$, the existence of algorithms which perform much better than the exhaustive search algorithms for these problems is unlikely.

The general techniques mentioned above do not apply to the case $t = 1$, and hence, do not imply computational lower bounds for $W[1]$-hard problems that are not (or are not known to be) $W[2]$-hard. We develop new techniques to derive computational lower bounds on $W[1]$-hard problems based on a stronger assumption. Consider the optimization class SNP introduced by Papadimitriou and Yannakakis [18], which consists of all search problems expressible by second-order existential formulas whose first-order part is universal. Impagliazzo and Paturi [14] introduced the notion of SERF-completeness for the class SNP and identified a class of problems which is complete for SNP under SERF-reduction, such that the subexponential time solvability for any of these SERF-complete problems implies that all SNP problems are solvable in subexponential time. The class SNP contains many well-known NP-hard problems, 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 [23].

Therefore it seems convincing to assume that not all SNP problems are solvable in subexpo-

[1]An immediate question that might come to mind is whether such a $W[1]$-hard problem exists. The answer is affirmative: by re-defining the parameter, it is not difficult to construct $W[1]$-hard problems that are solvable in time $O(n^{\lg \lg k})$.

nential time. This assumption actually implies $W[1] \neq \text{FPT}$ [1]. Under this stronger assumption, all computational lower bounds we derive for the $W[2]$-hard problems mentioned above still hold. Moreover, under this stronger assumption, we are now able to derive computational lower bounds for certain $W[1]$-hard parameterized problems that are not (or not known to be) $W[2]$-hard: we prove that a group of problems including the following problems cannot be solved in time $n^{o(k)}$: WEIGHTED q-SAT (for any fixed $q \geq 2$), INDEPENDENT SET, and CLIQUE. Again these lower bounds prove that one cannot expect to have algorithms that perform much better than the exhaustive search algorithms for these problems.

Before we close this section, we point out that besides the area of parameterized complexity theory, the topics related to the above research have also been investigated from many other angles. Exact algorithms that provide computational upper bounds for NP-hard problems have been extensively studied in the last three decades (see survey [23] and its references). For computational lower bounds, Papadimitriou and Yannakakis [19] introduced the class LOGSNP. Problems in this class specifically set (or implicitly force) the parameter value k to be equal to $\lg n$. A group of problems for which no polynomial time algorithms are known, such as log-CLIQUE, TOURNAMENT DOMINATING SET, and VC-DIMENSION, are proved to be LOGSNP-hard, in the sense that if any of these problems can be solved in polynomial time then all problems in LOGSNP are solvable in polynomial time. Thus, LOGSNP-hardness provides a computational lower bound for these problems. Feige and Kilian [12] studied the computational complexity of finding a clique of size $k = \lg n$, and showed that if this problem can be solved in polynomial time then nondeterministic computation can be simulated by deterministic computation in subexponential time. Moreover, they showed that if the k-CLIQUE problem, where $k = \Theta(\lg^c n)$ for some constant c, can be solved in time $O(n^h)$, where $h = k^{1-\epsilon}$ for some $\epsilon > 0$, then nondeterministic circuits can be simulated by randomized or non-uniform deterministic circuits of subexponential size (see [12] for the formal definitions). Compared to this result, our results on CLIQUE have a weaker assumption (i.e., CLIQUE can be solved in time $n^{o(k)}$) but our conclusion (i.e., all problems in SNP are subexponential time solvable) is not necessarily weaker.

2 Preliminaries

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. Therefore, 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* [11] if there is a (parameterized) algorithm that decides whether an input (x, k) is a member of Q in time $O(f(k)|x|^c)$, where c is a fixed constant and $f(k)$ is a recursive function independent of the input length $|x|$. Let FPT denote the class of all fixed-parameter tractable problems.

To study the fixed-parameter tractability, the *fpt-reduction* has been introduced [11]: 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 $O(f(k)|x|^c)$, where f and g are recursive 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 0th level of the hierarchy lies the class FPT, and at the ith level, the class $W[i]$ for $i > 0$. 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 $W[i]$. If any $W[i]$-hard problem is in

FPT, then $W[i] = \text{FPT}$, which, to the common belief, is very unlikely [11].

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. A circuit is said to be *monotone* (resp. *antimonotone*) if all its input literals are positive (resp. negative). A circuit represents a Boolean function in a natural way. Using the results in [6], 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. The *depth* of a node v is i if v is at the ith level. The *depth* of a circuit is d if its output gate has depth d. A circuit is a Π-*circuit* if its output gate is an AND gate, and is a Π_h-*circuit* if it has depth h and its output gate is an AND gate. 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. The *weight* of an assignment τ is the number of variables assigned value 1 by τ. A propositional formula F is said to be t-*normalized* where $t \geq 1$, if F is the products-of-sums-of-products ... of literals with t alternations [11]. From a t-normalized formula F with n input variables, we can naturally correspond an equivalent Π_t circuit C_F with n input variables.

The t-NORMALIZED SATISFIABILITY problem where $t \geq 1$, abbreviated SAT$[t]$ henceforth, is defined as follows: Given a t-normalized formula F over n variables whose size is m, decide if F is satisfiable. For instance, the 2-NORMALIZED SATISFIABILITY problem is the same as the satisfiability problem SAT. The WEIGHTED t-NORMALIZED SATISFIABILITY problem is defined as follows: Given a t-normalized formula F over n variables whose size is m, and a positive integer k, decide if F has a satisfying assignment of weight k. From the WEIGHTED t-NORMALIZED SATISFIABILITY problem we can define the WEIGHTED MONOTONE t-NORMALIZED SATISFIABILITY problem (resp. WEIGHTED ANTIMONOTONE t-NORMALIZED SATISFIABILITY problem) by requiring all the input literals of the formula to be positive (resp. negative) [11]. It is known that for $t > 1$, the WEIGHTED t-NORMALIZED SATISFIABILITY problem, the WEIGHTED MONOTONE t-NORMALIZED SATISFIABILITY problem for even t, and the WEIGHTED ANTIMONOTONE t-NORMALIZED SATISFIABILITY problem for odd t, are all $W[t]$-complete [11]. Also, it is known that the WEIGHTED ANTIMONOTONE 2-SATISFIABILITY (each clause contains at most two literals) is $W[1]$-complete [11].

The above problems can be naturally extended to the circuit model. The WEIGHTED t-NORMALIZED SATISFIABILITY corresponds to the WEIGHTED t-NORMALIZED CIRCUIT SATISFIABILITY, abbreviated WCS$[t]$: Given a Π_t circuit C with n input variables whose size is m, and a positive integer k, decide if C has a satisfying assignment of weight k. Similarly, the WEIGHTED MONOTONE t-NORMALIZED SATISFIABILITY corresponds to the WEIGHTED MONOTONE t-NORMALIZED CIRCUIT SATISFIABILITY, abbreviated MONOTONE WCS$[t]$, and the WEIGHTED ANTIMONOTONE t-NORMALIZED SATISFIABILITY problem corresponds to the WEIGHTED ANTIMONOTONE t-NORMALIZED CIRCUIT SATISFIABILITY, abbreviated ANTIMONOTONE WCS$[t]$. Since these problems are natural extensions of the above problems to the circuit model, and since every t-normalized formula can be transformed into an equivalent Π_t circuit with the same number and monotonicity of the input variables, and with no more than a polynomial increase in the size, we have WCS$[t]$, MONOTONE WCS$[t]$ for even t, and ANTIMONOTONE WCS$[t]$ for odd t, where $t > 1$, are all $W[t]$-complete. Moreover, ANTIMONOTONE 2-WCS$[2]$, where each gate at level 1 is required to have fan-in bounded by two, is $W[1]$-complete.

3 Lower bounds for the W-hierarchy

In this section we give lower bounds on the time complexity of the WCS$[t]$ problem, where $t \geq 2$, which is complete for the class $W[t]$ in the W-hierarchy. These lower bounds directly imply lower bounds for many natural parameterized problems including WEIGHTED SAT, DOMINATING SET, HITTING SET, and SET COVER. We will break our main theorem into two intermediate theorems. We shall present both theorems because these theorems, per se, are of interest.

Theorem 3.1 *For any $t \geq 2$, if* WCS$[t]$ *can be solved in time $n^{o(k)}h(m)$ then* SAT$[t]$ *can be solved in time $2^{o(n)}h'(m)$, where h and h' are two polynomials.*

PROOF. Fix $t \geq 2$, and suppose that WCS$[t]$ can be solved in time $n^{o(k)}h(m)$. This means that there exists an unbounded non-decreasing function $r(k)$, such that WCS$[t]$ can be decided in time bounded by $n^{k/r(k)}h(m)$. We will show that SAT$[t]$ can be solved in time $2^{o(n)}h'(m)$. Let F be a t-normalized formula over n variables and of size m. We can naturally associate an equivalent Π_t circuit C_F with n input variables, whose size is polynomial in m, such that F is satisfiable if and only if C_F is. Note also that C_F can be constructed from F in time polynomial in m. We will construct an instance (C'_F, k) of WCS$[t]$ such that C_F is satisfiable if and only if C'_F has a satisfying assignment of weight k. The construction distinguishes two cases depending on the parity of t.

Case 1. t is even. In this case the gates in level 1 of C_F are OR gates. Suppose that $x_1, \ldots, x_n$ are the input variables to C_F. Let $r = \lfloor \lg n \rfloor$, $b = \lceil n/r \rceil$, and $s = 2^r$. We divide the n input variables $x_1, \ldots, x_n$ into b blocks $B_1, \ldots, B_b$, where block B_i consists of input variables $x_{(i-1)r+1}, \ldots, x_{ir}$, for $i = 1, \ldots, b-1$, and B_b consists of input variables $x_{(b-1)r+1}, \ldots, x_n$. Denote by the size of a block the number of variables in the block, and note that $|B_i|$, $i = 1, \ldots, b-1$, is exactly r, and $|B_b| = r' \leq r$. We form the b blocks $B'_1, \ldots, B'_b$, each block B'_i, $i = 1, \ldots, b-1$, consists of exactly s new variables $z_i^1, \ldots, z_i^s$, and B'_b consists of $s' = 2^{r'}$ variables $z_b^1, \ldots, z_b^{s'}$. Blocks $B'_1, \ldots, B'_b$ will be used to decode the input variables in blocks $B_1, \ldots, B_b$ in the following manner. The input variable z_i^j in block B'_i, $i = 1, \ldots, b$, $j = 1, \ldots, |B'_i|$, will be used to indicate an assignment to the input variables in B_i such that if $z_i^j = 1$, then the variables in B_i will be assigned the bits in the binary representation of the number j. Since for every $i = 1, \ldots, b$, we have $|B'_i| = 2^{|B_i|}$, it is clear that there is a bijection between the assignments to the variables in B'_i of weight 1 (with respect to the variables in B'_i) and the possible binary configurations of the input variables in B_i, given by the above description. It follows that there is a bijection between all possible truth assignments to the input variables in C_F, and all truth assignments to the input variables in B'_i, $i = 1, \ldots, b$, in which exactly one variable in each block B'_i is assigned the value 1. The circuit C'_F is now constructed from C_F by removing the input literals of C_F and adding the new input variables in the blocks $B'_1, \ldots, B'_b$. The new input variables are connected to the OR gates at level 1 in C_F as follows. Suppose that the positive (resp. negative) literal l corresponding to variable x_q ($q \in \{1, \ldots, n\}$) is connected to an OR gate g at level 1 in C_F, and suppose that x_q is the pth variable in block B_i, $i \in \{1, \ldots, b\}$. Then all input variables z_i^j, $j = 1, \ldots, |B'_i|$, in block B'_i such that the pth bit in the binary representation of j is 1, are connected to gate g. We also add some "enforcement" circuitry to C'_F to ensure that at least one new variable z_i^j in every block B'_i, $i = 1, \ldots, b$, $j = 1, \ldots, |B'_i|$, is set to 1. This can be achieved as follows. For every block B'_i where $i = 1, \ldots, b$: add an OR gate g'_i, connect every variable in B'_i to g'_i, and connect g'_i to the output gate of C_F. This completes the construction of C'_F. Clearly, C'_F has size $h''(m)$ for some polynomial h'', and can be constructed from C_F in time polynomial in m. Moreover, since $t \geq 2$, and the enforcement circuitry requires no more than depth 2 to be implemented, C'_F is also a Π_t circuit. It is not difficult to verify that F is satisfiable if and only if C_F is satisfiable, if and only if C'_F has a satisfying assignment of weight b. Note that any satisfying assignment to C'_F of weight b must satisfy exactly one input variable in each block B'_i, $i = 1, \ldots, b$. The reason being that the number of blocks is exactly b, and the enforcement circuitry guarantees that at least one variable in every block is set to 1 in any satisfying assignment. Since C'_F is a Π_t circuit, it follows that (C'_F, b) is an instance of WCS$[t]$. The number of input variables N to C'_F is bounded by $b.s = \lceil n/r \rceil . 2^{\lfloor \lg n \rfloor} \leq n^2$. The parameter k in the instance (C'_F, b) is equal to $b = \lceil n/r \rceil = \lceil n/\lfloor \lg n \rfloor \rceil \leq 2n/\lg n$ when n is large enough (if n is bounded by a constant the problem can be solved in constant time). By the hypothesis, we can decide $(C'_F, k = b)$ in time bounded by $N^{k/r(k)} h(h''(m)) \leq n^{4n/r'(n)\lg n} h(h''(m))$, where $r'(n) = r(\lceil n/\lfloor \lg n \rfloor \rceil)$ is an unbounded non-decreasing function of n. Since $n^{4n/r'(n)\lg n} \in 2^{o(n)}$, and since the construction of C_F and C'_F from F can be done in polynomial time in m, it follows that deciding whether C'_F has a truth assignment of weight k, and hence whether F is satisfiable, can be done in time $2^{o(n)} h'(m)$, where h' is a polynomial.

Case 2. t is odd. Since $t > 1$, $t \geq 3$, and the gates at level 1 in C_F are AND gates. The decomposition of the n input variables in C_F into b blocks B_i, $i = 1, \ldots, b$, and the construction of the blocks B'_i, proceed exactly as in **Case 1**. The enforcement circuitry which ensures that exactly one variable z_i^j in block B'_i is set to true also remains the same. Since $t \geq 3$, this enforcement circuitry can still be implemented without affecting the level structure of C_F (this enforcement circuitry needs two levels to be implemented: AND-of-OR's). The only part in the construction of C'_F that is different from the above construction, and is a bit trickier, is how to connect the new variables and their negations to the AND gates at level 1 in C_F. Let g be a level-1 AND gate in C_F. Let S_i, $i = 1, \ldots, b$ be the set of literals connected to g whose variables are in block B_i (some S_i's may be empty). If g is satisfied, then all literals in S_i, $i = 1, \ldots, b$, must receive the value 1. Let S'_i, $i = 1, \ldots, b$, be the set of variables z_i^j in block B'_i, such that if the input variables in B_i are assigned the corresponding bits in the binary representation of j (i.e., the pth input variable in B_i is assigned the pth bit in the binary representation of j), all literals in S_i receive the value 1. Let $S''_i = \{z_i^j \in B'_i \mid z_i^j \notin S'_i\}$, and $\overline{S''_i} = \{\overline{z}_i^j \mid z_i^j \in S''_i\}$, where $\overline{z}_i^j$ is the negation of the input variable z_i^j. For every level-1 gate g in C_F, and for every $i = 1, \ldots, b$, we remove the input literals from S_i to gate g in C_F, and connect the literals in $\overline{S''_i}$ to g. Let C'_F be the resulting circuit. Then clearly C'_F is a Π_t circuit. We argue next that the bijection described above between truth assignments to the input variables in C_F and those to the input variables in C'_F that assign exactly one variable in every block B'_i, $i = 1, \ldots, b$, the value 1, associates with every truth assignment τ to C_F a truth assignment τ' to C'_F such that τ satisfies C_F if and only if τ' satisfies C'_F. The only difference between C_F and C'_F is the input gates and their connections to level-1 gates. So it suffices to argue that the truth value of a level-1 gate g in C_F with respect to τ is the same as its truth value in C'_F with respect to τ'. Now gate g receives the value 1 by τ in C_F if and only if all literals in S_i, $i = 1, \ldots, b$, are set to 1 by τ. This is true if and only if one of the variables in S'_i is set to 1 by τ' in C'_F. Since τ' assigns exactly one variable in every block the value 1, the latter condition is true if and only if all the variables in S''_i receive the value 0 by τ' in C'_F, which is in turn true if and only if all literals in $\overline{S''_i}$ receive the value 1 by τ' in C'_F. This is true if and only if g receives the value 1 by τ' in C'_F. The proof from this point on proceeds in exactly the same fashion as in **Case 1**.

We conclude that SAT$[t]$ can be decided in $2^{o(n)} h'(m)$ time. This completes the proof. $\square$

Theorem 3.2 *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.*

PROOF. If $t = 2$, the theorem states that if SAT can be solved in time $2^{o(n)}h(m)$ then $W[1]$=FPT. This result was established by Cai and Juedes [4]. 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 MONOTONE 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 MONOTONE 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 $x_1, \ldots, x_n$ be the input variables to C. We will construct a circuit C' from C with $k\lceil \lg n \rceil$ 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 $r = \lceil \lg n \rceil$ 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 level-1 OR gate in C. Let x_p be an input to g, and let $b_1b_2\ldots b_r$ 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 $g_p^1, \ldots, g_p^k$. 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 contains the binary representation of p, and hence, encodes x_p. The input to gate g_p^i is determined as follows. For $j = 1, \ldots, r$, 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 $g_p^i, i = 1, \ldots, k$, to g. We repeat this process for every level-1 gate g in C and every input variable in $\{x_1, \ldots, x_n\}$ 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, $i = 1, \ldots, k$, encode distinct k variables. This can be easily 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 SAT$[t]$ with kr input variables. It follows that we can decide if C' is satisfiable in time bounded by $T(n) = 2^{kr/s(kr)}h(h'(m)) = 2^{k\lceil \lg n \rceil/s(k\lceil \lg n \rceil)}h(h'(m)) \leq 2^{k(\lg n+1)/s'(n)}h''(m)$, for some unbounded non-decreasing function $s'(n)$, and some polynomial h''. Thus $T(n) \in 2^{o(\lg n)k}h''(m)$, and WCS$[t-1]$ is solvable in time $2^{o(\lg n)k}h''(m)$ for some polynomial h''. It follows that WCS$[t-1]$ is fixed parameter tractable (see Lemma 2.1 in [4] for a proof of this fact), and hence, $W[t-1]$ = FPT.

Case 2. t is even, and hence $t-1 \geq 3$ is odd. We consider the ANTIMONOTONE 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 ANTIMONOTONE 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 $W[t-1] = \text{FPT}$. This completes the proof. □

Theorem 3.3 *For any $t \geq 2$, if* WCS$[t]$ *is solvable in time $n^{o(k)}p(m)$ for some polynomial p, then $W[t-1] = FPT$.*

PROOF. The proof follows immediately from Theorem 3.1 and Theorem 3.2. □

Theorem 3.4 *If* MONOTONE WCS$[2]$ *is solvable in time $n^{o(k)}p(m)$ for some polynomial p, then $W[1] = FPT$.*

PROOF. Observe that when t is even, the circuit C'_F in the proof of Theorem 3.1 is monotone. Combining this observation with Theorem 3.1 and Theorem 3.2, the result follows. □

We define the following $W[2]$-complete problems [9, 11].

WEIGHTED SAT

Given a formula F in CNF over N variables whose size is m, decide if there exists a weight-k satisfying assignment for F.

RED/BLUE DOMINATING SET

Given a bipartite graph $G = (V, E)$, where $V = V_{red} \cup V_{blue}$ with $|V_{red}| = N$, and a positive integer k, decide if there exists a subset $V' \subseteq V_{red}$ of cardinality k such that V' dominates V_{blue} (i.e., every vertex in V_{blue} is adjacent to some vertex in V').

HITTING SET

Given a set $U = \{u_1, \ldots, u_N\}$, a set $S = \{S_1, \ldots, S_m\}$, where $S_i \subseteq U$, $i = 1, \ldots, m$, and a positive integer k, decide if there exists a subset $H \subseteq U$ with cardinality k, such that $H \cap S_i \neq \emptyset$ for $i = 1, \ldots, m$.

SET COVER

Given a set S, a collection $\mathcal{F} = \{C_1, \ldots, C_N\}$ of subsets of S such that $\bigcup_{i=1}^{N} C_i = S$, and a positive integer k, decide if there is a subset $\mathcal{C}$ of $\mathcal{F}$ with cardinality k such that $\bigcup_{C_i \in \mathcal{C}} C_i = S$.

FEATURE SET

Given a set of m examples $X = \{x^{(1)}, \ldots, x^{(m)}\}$, where $x^{(i)} = (x_1^{(i)}, \ldots, x_N^{(i)}, t^{(i)}) \in \{0,1\}^{N+1}$, and an integer $k > 0$, decide if there exists a *feature set* $S \subseteq \{1, \ldots, N\}$ of cardinality k, such that for all pairs of examples $i \neq j$, if $t^{(i)} \neq t^{(j)}$, then there exists $l \in S$ with $x_l^{(i)} \neq x_l^{(j)}$.

Theorem 3.5 *If any of the* WEIGHTED SAT*,* RED/BLUE DOMINATING SET*,* HITTING SET*,* SET COVER*, or* FEATURE SET *problems can be solved in time $N^{o(k)}p(|I|)$, then $W[1] = FPT$, where $|I|$ is the input size, N is the size of the universal set from which the k-element solution is to be chosen, and p is a polynomial.*

PROOF. The result for WEIGHTED SAT follows directly from Theorem 3.1 with $t = 2$, by observing that WEIGHTED SAT is exactly WEIGHTED 2-NORMALIZED SATISFIABILITY, which corresponds to WCS[2]. We show the result for RED/BLUE DOMINATING SET next. By Theorem 3.4, it suffices to show that if RED/BLUE DOMINATING SET is solvable in time $N^{o(k)}p(|G|)$, where G is the input graph and $N = |V_{red}|$, then MONOTONE WCS[2] is solvable in time $n^{o(k)}q(m)$ where n is the number of input variables, and m is the circuit size. Let (C, k) be an instance of MONOTONE WCS[2] where C has n input variable $\{x_1, \ldots, x_n\}$, and size m. Observe that C has a weight-k satisfying assignment if an only if there exists a weight-k assignment to the input variables in C such that all level-1 gates in C are satisfied. The last statement is true if and only if each level-1 gate in C has at least one variable of value 1 that is connected to it (since all level-1 gates in C are OR gates). Let $g_1, \ldots, g_r$ be the level-1 gates in C. We construct the bipartite graph $G = (V_{red} \cup V_{blue}, E)$ as follows. For each input variable x_i, $i = 1, \ldots, n$, we associate a vertex $x_i \in V_{red}$. For each gate g_j, $j = 1, \ldots, r$, we associate a vertex g_j in V_{blue}. Now a vertex x_i, $i = 1, \ldots, n$, is connected to a vertex g_j, $j = 1, \ldots, r$ in G if and only if x_i is an input to gate g_j in C. From the above discussion, it is easy to see that C has a weight-k satisfying assignment if and only if G has a subset $V' \subseteq V_{red}$ with $|V'| = k$, such that V' dominates V_{blue}. Thus, by solving the instance (G, k) of the RED/BLUE DOMINATING SET we can solve the instance (C, k) of MONOTONE WCS[2]. Since the construction of G from C can be done in time polynomial in m, it follows that if RED/BLUE DOMINATING SET can be solved in time $N^{o(k)}p(|G|)$, then MONOTONE WCS[2] can be solved in time $n^{o(k)}q(m)$ for some polynomial q (note that $N = n$ and $|G|$ is polynomial in m).

Now we show the result for the HITTING SET problem. It suffices to show that if HITTING SET can be solved in time $N^{o(k)}p(|I|)$ then RED/BLUE DOMINATING SET can be solved in time $n^{o(k)}q(|I'|)$, where N is the number of elements in the universe U, $|I|$ the size of the input instance I of HITTING SET, n the number of vertices in V_{red}, $|I'|$ the size of the input instance I' of RED/BLUE DOMINATING SET, and p and q are two polynomials. Let $I' = (G = ((V_{red} \cup V_{blue}), E), k)$, be an instance of RED/BLUE DOMINATING SET, where $|V_{red}| = n$, we construct and instance $I = ((U =$

$\{u_1, \ldots, u_N\}, S = \{S_1, \ldots, S_m\}), k)$ of HITTING SET as follows. The elements in U are the vertices in V_{red}. For every vertex v_j, $j = 1, \ldots, m$, in V_{blue} we associate a set S_j consisting of all neighbors of v_j. It is easy to see that $|I|$ is polynomial in the $|I'|$, and that the construction can be carried out in time polynomial in $|I'|$. It can be easily verified that S has a hitting set of size k if and only G has a subset $V' \subseteq V_{red}$ of k vertices that dominates V_{blue}. Noting that $N = n$, the statement follows.

To show that the same result holds for SET COVER we reduce HITTING SET to SET COVER. Let $I = ((U = \{u_1, \ldots, u_N\}, S = \{S_1, \ldots, S_m\}), k)$ be an instance of HITTING SET, we construct an instance $I' = ((S', \mathcal{F}), k)$ of SET COVER as follows: $S' = \{S_1, \ldots, S_m\}$ and $\mathcal{F} = \{C_1, \ldots, C_N\}$ where $C_i = \{S_j \in S' \mid u_i \in S_j\}$. It is not difficult to see that S has a hitting set of size k if and only if S' has a set cover of size k. Noting that I' has size polynomial in I, and the construction of I' from I takes polynomial time in $|I|$, the statement follows.

Finally, the result for the FEATURE SET problem follows from the reduction from SET COVER to FEATURE SET given in [22]. □

We can prove a similar result for the $W[2]$-complete problem DOMINATING SET (given a graph G and an integer k, decide whether there is a subset D of k vertices such that every vertex not in D is adjacent to at least one vertex in D) after applying a polynomial time pre-processing algorithm to the input instances. The pre-processing algorithm, which runs in polynomial time, reduces the set of vertices V to a subset $U \subseteq V$, with cardinality N, from which the k vertices are to be chosen.

Suppose we are looking for a dominating set of k vertices in a graph G of n vertices. Consider any independent set I of $k+1$ vertices in G. We say I has *the same neighbor set* A if every vertex in I has A as its neighbor set. Obviously, if G has a dominating set of k vertices and I has the same neighbor set A, then at least one of the vertices in A must be included in the dominating set. Thus, the set A can be identified as part of the search space U. This suggests the following pre-processing algorithm to an instance (G, k) of the DOMINATING SET problem:

DS-Core

```
1. $i = 0$;  color all vertices in $G$ WHITE;
2. while there is an independent set $I_i$ of $k+1$ WHITE vertices with the same neighbor set $A_i$
       color all vertices in $A_i$ BLACK;
       color each vertex not in $A_i$ but adjacent to a vertex in $A_i$ RED;
       $i = i + 1$;
3. if $i < k$, let $U$ be the set of all vertices in $G$; stop.
4. if $i = k$, let $U = \cup_{i=0}^{k-1} A_i$, stop.
5. if $i > k$, stop ($G$ has no dominating set of $k$ vertices).
```

It is easy to verify the following facts about the above algorithm: (1) all independent sets I_i are disjoint; (2) no two vertices in two different independent sets I_i and I_j, $i < j$, share a common neighbor (otherwise, the construction of I_i would have colored a vertex in I_j RED, which in turn would have not been included in I_j). Therefore, each vertex set A_i must contain at least one vertex in any dominating set of k vertices in the graph G, if such a dominating set exists. Thus, if there are more than k such sets A_i, we can directly conclude the nonexistence of such a dominating set; while if there are exactly k such subsets A_i, we can simply concentrate on the set $\cup_{i=0}^{k-1} A_i$ to see if it contains a dominating set of k vertices for the entire graph G. In case there are less than k

such subsets A_i, the pre-processing algorithm simply does not help and we reset the search space U to be the entire vertex set of G. Finally, note that in the adjacency matrix representation of the graph G, an independent set I_i of $k+1$ WHITE vertices with the same neighbor set corresponds to $k+1$ identical rows in the adjacency matrix, which can be easily identified in polynomial time. Thus, the algorithm **DS-Core** runs in polynomial time.

Theorem 3.6 *If the* DOMINATING SET *problem can be solved in time $N^{o(k)}p(|G|)$, then $W[1] = FPT$, where $N \leq |V(G)|$, is the size of the universal set $U \subseteq V(G)$ obtained by applying the algorithm* **DS-Core** *to G, and p is a polynomial.*

PROOF. To prove the theorem, it suffices to show that if the DOMINATING SET problem is solvable in time $N^{o(k)}p(|G|)$, where N is the size of the universal set U obtained by applying the algorithm **DS-Core**, then WEIGHTED SAT is solvable in time $n^{o(k)}q(m)$ for some polynomial q. This, together with Theorem 3.5, will give the result.
We use the polynomial time reduction given in [11] from WEIGHTED SAT described as follows. From a CNF formula F of n variables and m clauses, a graph G of $O(n^3+m)$ edges is constructed such that F has a satisfying assignment of weight k if and only if G has a dominating set of $2k$ vertices. A more careful examination of the reduction reveals that in the graph G, there are $2k$ disjoint independent sets I_i, $i = 1, \ldots, 2k$, where each I_i has $2k+1$ vertices and has the same neighbor set A_i consisting of $O(n^2)$ vertices, such that all neighbor sets A_i are also disjoint. Moreover, besides these independent sets, no two vertices in G share the same neighbor set. Therefore, if we apply the algorithm **DS-Core** to G, the constructed search space U will be exactly the union of these $2k$ neighbor sets $A_1, \ldots, A_{2k}$. Thus, for the graph G, the size N of the search space U is $O(kn^2)$. Using this polynomial time reduction, it is easy to verify from the above information that if the DOMINATING SET problem is solvable in time $N^{o(k)}p(|G|)$, then this will imply that WEIGHTED SAT is solvable in $n^{o(k)}q(m)$ time for some polynomial q. This completes the proof. $\square$

4 Lower bounds for some $W[1]$-complete problems

In this section we prove that the existence of $n^{o(k)}$ time algorithms for many parameterized problems like INDEPENDENT SET, CLIQUE, and WEIGHTED q-SAT implies that all problems in the class SNP can be solved in subexponential time. The class SNP [18] contains many well-known NP-hard problems including q-SAT, q-COLORABILITY, q-SET COVER, VERTEX COVER, and INDEPENDENT SET [14]. It is commonly believed that it is unlikely that all problems in SNP are solvable in subexponential time[2]. We start with the following theorem, which is due to Nemhauser and Trotter [16]. This version of the theorem appears in [7].

Theorem 4.1 *([7]) Given an instance (G, k) of* VERTEX COVER*, there is a polynomial time algorithm which either reports that G does not have a vertex cover of size k, or produces a subgraph G' of G with at most $2k'$ vertices, where $k' \leq k$, such that G has a vertex cover of size k if and only if G' has a vertex cover of size k'.*

Theorem 4.2 *If the parameterized* INDEPENDENT SET *problem can be solved in time $n^{o(k)}$, where n is the number of vertices in the graph, then all problems in SNP can be solved in subexponential time.*

[2]A recent result showed the equivalence between the statement that all SNP problems are solvable in subexponential time, and the collapse of a parameterized class, called $Mini[1]$, to FPT [10].

Proof. 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 $O(n^{f(k)})$ steps, where $f(k) \leq k/r(k)$ for some unbounded nondecreasing function $r(k)$. We will show that the VERTEX COVER problem can be solved in time $2^{o(k)}p(n)$, for some polynomial p (note that this will imply that VERTEX VOVER can be solved in time $2^{o(n)}$, which is subexponential). Since the VERTEX COVER problem is complete for the class SNP under SERF reductions, this will show that all problems in SNP can be solved in subexponential time [14].

Let $(G = (V, E), k)$ be an instance of VERTEX COVER. By Theorem 4.1, we can assume that G has at most $n \leq 2k$ vertices. We partition the n vertices of G into $k' = \lceil \frac{n}{\lceil \log k \rceil} \rceil$ blocks $B_1, B_2, \ldots, B_{k'}$ each of size bounded by $\lceil \lg k \rceil$. 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 $|B_i| \leq \lceil \lg k \rceil$, this approach converts the single question "does G have a vertex cover of size k?" into at most

$$\begin{aligned}
\lceil \lg k \rceil^{k'} &\leq \lceil \lg k \rceil^{\lceil \frac{2k}{\lceil \lg k \rceil} \rceil} \\
&= 2^{\lceil \frac{2k}{\lceil \lg k \rceil} \rceil \cdot \lg (\lceil \lg k \rceil)} \\
&= 2^{o(k)}
\end{aligned}$$

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 a most $2k \cdot k' \leq 4k^2$ vertices because $|B_i| \leq \lceil \lg k \rceil$, and there are at most $\binom{\lceil \lg k \rceil}{k_i} \leq 2^{\lceil \lg k \rceil} \leq 2^{\lg k + 1} \leq 2k$, possible subsets B_{ij} of size k_i. Therefore, applying algorithm A to the instance (G^*, k') takes

$$\begin{aligned}
(4k^2)^{f(k')} &= 2^{f(k') \cdot \lg(4k^2)} \\
&\leq 2^{k'/r(k')(2+2\lg k)} \\
&\leq 2^{(2k/(s(k)\lg k) + 1/s(k))(2+2\lg k)} \\
&= 2^{o(k)}
\end{aligned}$$

where $s(k) = r(k')$ is an unbounded non-decreasing function of k. The inequality before the last uses the fact that $k' = \lceil \frac{n}{\lceil \log k \rceil} \rceil \leq 2k/\lg k + 1$.
Noting that the time needed to construct G^* is $O(k^4)$, and that applying Theorem 4.1 takes polynomial time in n, it follows that the VERTEX COVER problem can be solved in time $q(n) + 2^{o(k)} \cdot 2^{o(k)} \cdot k^4 \leq 2^{o(k)} p(n)$, where p and q are polynomials. This completes the proof. □

Consider the following parameterized problem.

> WEIGHTED q-SAT:
>
> Given a CNF formula F on n variables with at most q literals per clause, where $q \geq 2$, determine if there is a weight-k assignment to the variables that satisfies F.

Theorem 4.3 *If* CLIQUE, *or* WEIGHTED q-SAT *(for $q \geq 2$), can be solved in time $n^{o(k)}$ then all problems in SNP can be solved in subexponential time.*

PROOF. It is well-known that a graph G with n vertices has a clique of size k if and only if the complement of G, $\overline{G}$, has an independent set of size k. Hence, if CLIQUE has a $n^{o(k)}$ time algorithm, then parameterized INDEPENDENT SET has a $n^{o(k)}$ time algorithm. Applying Theorem 4.2 completes the proof. Similarly, the reduction from INDEPENDENT SET to WEIGHTED 2-SAT is straightforward. Given a graph G with n vertices and an integer k, we can convert the instance (G, k) of INDEPENDENT SET to an instance (F, k) of WEIGHTED 2-SAT as follows. For each vertex v_i in G, create a Boolean variable x_i. For each edge $(v_i, v_j) \in E$, create the clause $(\neg x_i \vee \neg x_j)$. The formula F created by taking the conjunction of all these clauses, is an instance of WEIGHTED 2-SAT with at most $O(n^2)$ clauses. Moreover, F has a weight-k satisfying assignment if and only if G has an independent set of size k. Hence, if WEIGHTED 2-SAT has an $n^{o(k)}$ time algorithm, then INDEPENDENT SET can be solved in $n^{o(k)}$ time. □

Note that all the above problems are $W[1]$-complete. Also note that similar results to the ones in this section and the previous one, can be derived for many other $W[1]$-hard and $W[2]$-hard problems.

References

[1] K. A. Abrahamson, R. G. Downey, and M. R. Fellows, Fixed-parameter tractability and completeness IV: on completeness for $W[P]$ and PSPACE analogs, *Annals of Pure and Applied Logic 73*, pp. 235-276, (1995).

[2] J. Alber, H. L. Bodlaender, H. Fernau, T. Kloks, R. Niedermeier, Fixed parameter algorithms for dominating set and related problems on planar graphs, *Algorithmica 33*, pp. 461-493, (2002).

[3] J. Buhler and M. Tompa, Finding motifs using random projections, *Journal of Computational Biology 9*, pp. 225-242, (2002).

[4] L. Cai and D. Juedes, On the existence of subexponential parameterized algorithms, *Journal of Computer and System Sciences*, to appear.

[5] J. Cheetham, F. Dehne, A. Rau-Chaplin, U. Stege, and P. Taillon, Solving large FPT problems on coarse grained parallel machines, *Journal of Computer and System Sciences*, to appear.

[6] J. Chen, Characterizing parallel hierarchies by reducibilities, *Information Processing Letters 39*, pp. 303-307, (1991).

[7] J. Chen, I. A. Kanj, and W. Jia, Vertex cover: further observations and further improvements, *Journal of Algorithms 41*, pp. 280-301, (2001).

[8] D. Coppersmith and S. Winograd, Matrix multiplication via arithmetic progression, *Journal of Symbolic Computation 9*, pp. 251-280, (1990).

[9] C. Cotta and P. Moscato, The k-feature set problem is $W[2]$-complete, *Journal of Computer and System Sciences*, (2002), to appear.

[10] R. Downey, V. Estivill-Castro, M. Fellows, E. Prieto-Rodriguez, and F. Rosamond, Cutting Up is Hard to Do: the Parameterized Complexity of k-Cut and Related Problems, *Electronic Notes in Theoretical Computer Science* **78**, pp. 205–218, (2003).

[11] R.G. Downey and M.R. Fellows, *Parameterized Complexity*, Springer-Verlag, 1999.

[12] U. Feige and J. Kilian, On limited versus polynomial nondeterminism, *Chicago Journal of Theoretical Computer Science*, (1997).

[13] M. Garey and D. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, H. Freeman, New York, 1979.

[14] R. Impagliazzo and R. Paturi, Which problems have strongly exponential complexity? *Journal of Computer and System Sciences 63*, pp. 512-530, (2001).

[15] C. Roth-Korostensky, *Algorithms for Building Multiple Sequence Alignments and Evolutionary Trees*, Ph.D. Thesis, No. 13550, ETH Zürich, 2000.

[16] G. L. Nemhauser and L. E. Trotter, Vertex packing: structural properties and algorithms, *Mathematical Programming* **8**, pp. 232-248, (1975).

[17] J. Nešetřil and S. Poljak, On the complexity of the subgraph problem, *Commentationes Mathematicae Universitatis Carolinae* **26 (2)**, pp. 415-419, (1985).

[18] C. H. Papadimitriou and M. Yannakakis, Optimization, approximation, and complexity classes, *Journal of Computer and System Sciences 43*, pp. 425-440, (1991).

[19] C. H. Papadimitriou and M. Yannakakis, On limited nondeterminism and the complexity of VC dimension, *Journal of Computer and System Sciences 53*, pp. 161-170, (1996).

[20] P. A. Pevzner and S.-H. Sze, Combinatorial approaches to finding subtle signals in DNA sequences, *Proc. 8th International Conference on Intelligent Systems for Molecular Biology*, pp. 269-278, (2000).

[21] U. Stege, *Resolving Conflicts from Problems in Computational Biology*, Ph.D. Thesis, No. 13364, ETH Zürich, 2000.

[22] K. Van Horn and T. Martinez, The Minimum Feature Set Problem, *Neural Networks* **7 (3)**, pp. 491-494, (1994).

[23] G. J. Woeginger, Exact algorithms for NP-hard problems: a survey, *Lecture Notes in Computer Science 2570*, pp. 185-207, Springer-Varlag (2003).