Available online at www.sciencedirect.com

ScienceDirect

Journal of Computer and System Sciences 72 (2006) 1346–1367

JOURNAL OF COMPUTER AND SYSTEM SCIENCES

www.elsevier.com/locate/jcss

Strong computational lower bounds via parameterized complexity ☆

Jianer Chen [a,b,*,1], Xiuzhen Huang [c,2], Iyad A. Kanj [d,3], Ge Xia [e,4]

[a] Department of Computer Science, Texas A&M University, College Station, TX 77843-3112, USA
[b] College of Information Science and Engineering, Central South University, Changsha, Hunan 410083, PR China
[c] Computer Science Department, Arkansas State University, State University, AR 72467, USA
[d] School of CTI, DePaul University, 243 S. Wabash Avenue, Chicago, IL 60604, USA
[e] Department of Computer Science, Lafayette College, Easton, PA 18042, USA

Received 11 January 2005; received in revised form 18 April 2006

Available online 24 May 2006

Abstract

We develop new techniques for deriving strong computational lower bounds for a class of well-known NP-hard problems. This class includes WEIGHTED SATISFIABILITY, DOMINATING SET, HITTING SET, SET COVER, CLIQUE, and INDEPENDENT SET. For example, although a trivial enumeration can easily test in time $O(n^k)$ if a given graph of n vertices has a clique of size k, we prove that unless an unlikely collapse occurs in parameterized complexity theory, the problem is not solvable in time $f(k)n^{o(k)}$ for *any* function f, even if we restrict the parameter values to be bounded by an arbitrarily small function of n. Under the same assumption, we prove that even if we restrict the parameter values k to be of the order $\Theta(\mu(n))$ for *any* reasonable function μ, no algorithm of running time $n^{o(k)}$ can test if a graph of n vertices has a clique of size k. Similar strong lower bounds on the computational complexity are also derived for other NP-hard problems in the above class. Our techniques can be further extended to derive computational lower bounds on polynomial time approximation schemes for NP-hard optimization problems. For example, we prove that the NP-hard DISTINGUISHING SUBSTRING SELECTION problem, for which a polynomial time approximation scheme has been recently developed, has no polynomial time approximation schemes of running time $f(1/\epsilon)n^{o(1/\epsilon)}$ for any function f unless an unlikely collapse occurs in parameterized complexity theory.

Keywords: Parameterized computation; Computational complexity; Lower bound; Clique; Polynomial time approximation scheme

☆ A preliminary version of this paper "Linear FPT reductions and computational lower bounds" was presented at The 36th ACM Symposium on Theory of Computing, STOC 2004, Chicago, June 13–15, 2004 (see [J. Chen, X. Huang, I. Kanj, G. Xia, Linear FPT reductions and computational lower bounds, in: Proc. 36th ACM Symposium on Theory of Computing, STOC '04, 2004, pp. 212–221]).

* Corresponding author.
E-mail addresses: chen@cs.tamu.edu (J. Chen), xzhuang@csm.astate.edu (X. Huang), ikanj@cs.depaul.edu (I.A. Kanj), gexia@cs.lafayette.edu (G. Xia).

1 Supported in part by USA National Science Foundation under Grants CCR-0311590 and CCF-0430683, and by China National Natural Science Foundation under Grants Nos. 60373083 and 60433020 while this author was at College of Information Science and Engineering, Central-South University, Changsha, Hunan 410083, PR China.
2 Supported in part by USA National Science Foundation under Grant CCR-0000206.
3 Supported in part by DePaul University Competitive Research Grant.
4 Supported in part by USA National Science Foundation under Grants CCR-0311590 and CCF-0430683.

0022-0000/$ – see front matter
doi:10.1016/j.jcss.2006.04.007

1. Introduction

Parameterized computation is a recently proposed approach dealing with intractable computational problems. By taking the advantages of the small or moderate values of a parameter k, fixed-parameter tractable algorithms, whose running time takes the form $f(k)n^{O(1)}$ for a function f, have been used to solve a variety of difficult computational problems in practice. For example, the parameterized algorithm of running time $O(1.286^k + kn)$ for VERTEX COVER [7] has been quite practical in its applications in the research of multiple sequence alignments [5].

The rich positive toolkit of novel techniques for designing efficient and practical parameterized algorithms is accompanied in the theory by a corresponding negative toolkit that supports a theory of parameterized intractability. The concept of $W[1]$-hardness has been introduced, and a large number of $W[1]$-hard parameterized problems have been identified [13]. Now it has become commonly accepted in parameterized complexity theory that no $W[1]$-hard problem can be solved in time $f(k)n^{O(1)}$ for any function f (i.e., $W[1] \neq FPT$) [13]. Examples include a recent result by Papadimitriou and Yannakakis [23], proving that the DATABASE QUERY EVALUATION problem is $W[1]$-hard. This hints that it is unlikely that the problem can be solved by an algorithm whose running time is of the form $f(k)n^{O(1)}$, thus excluding the possibility of a practical algorithm for the problem even if the parameter k (the size of the query) is small as in most practical cases.

Thus, the $W[1]$-hardness of a parameterized problem implies that any algorithm of running time $O(n^h)$ solving the problem *must* have h a function of the parameter k. However, this does not completely exclude the possibility that the problem may become feasible for small values of the parameter k. For instance, if the problem is solvable by an algorithm running in time $O(n^{\log\log k})$, then such an algorithm is still feasible for moderately small values of k.[5]

The above problem was recently tackled in [8], where, by setting $k = \sqrt{n/\log n}$, it was proven that any $n^{o(k)}$-time algorithms for a class of $W[1]$-hard parameterized problems, such as CLIQUE, would induce unlikely collapses in parameterized complexity theory. Thus, algorithms of uniform running time $n^{o(k)}$ for these problems are unlikely because of the special parameter value $k = \sqrt{n/\log n}$. This result, however, does not answer the following question: can the problems be solvable in time $n^{o(k)}$ for parameter values $k \neq \sqrt{n/\log n}$ such as $k = \log\log n$ or $k = n^{4/5}$? Note that one would anticipate that for an extreme range of the parameter values, better algorithms might be possible by taking the advantage of the parameter values. Moreover, the results in [8] does not exclude the possibility that the problems may be solvable in time $f(k)n^{o(k)}$ for a function f. Note that the complexity of computational problems with parameter values other than $\sqrt{n/\log n}$ has been an interesting topic in research. We mention Papadimitriou and Yannakakis's work [22] that introduces the classes LOGNP and LOGSNP to study the complexity of a class of problems whose parameter values are, either implicitly or explicitly, bounded by $O(\log n)$. Constructing a clique of size $\log n$ in a graph of n vertices is one of the main problems studied in [22]. Feige and Kilian [14] studied the complexity of finding a clique of size $\log n$, and showed that if this problem can be solved in polynomial time then nondeterministic computation can be simulated by deterministic computation in subexponential time. They also showed that if a clique of size $\log^c n$ can be constructed in time $O(n^h)$, where c is a constant and $h = \log^{c-\epsilon} n$ for some $\epsilon > 0$, then nondeterministic circuits can be simulated by randomized or nonuniform deterministic circuits of subexponential size.

In this paper, based on the framework of parameterized complexity theory, we develop new techniques and derive stronger computational lower bounds for a class of well-known NP-hard problems. In particular, we answer the above mentioned questions completely. We start by proving computational lower bounds for a class of SATISFIABILITY problems, and then extend the lower bound results to other well-known NP-hard problems by introducing the concept of *linear fpt-reductions*. In particular, we consider two classes of parameterized problems: Class A which includes WEIGHTED CNF SAT, DOMINATING SET, HITTING SET, and SET COVER, and Class B which includes WEIGHTED CNF q-SAT for any constant $q \geqslant 2$, CLIQUE, and INDEPENDENT SET. We prove that (1) unless $W[1] = FPT$, no problem in Class A can be solved in time $f(k)n^{o(k)}m^{O(1)}$ for *any* function f, where n is the size of the search space from which the k elements are selected and m is the input length; and (2) unless all search problems in the syntactic class SNP introduced by Papadimitriou and Yannakakis [21] are solvable in subexponential time, no problem in Class B can be solved in time $f(k)m^{o(k)}$ for *any* function f, where m is the input length. These results remain true even if we bound the parameter values by an arbitrarily small nondecreasing and unbounded function. Moreover, under the same

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

assumptions, we prove that even if we restrict the parameter values k to be of the order $\Theta(\mu(n))$ for *any* reasonable function μ, no problem in Class A can be solved in time $n^{o(k)}m^{O(1)}$ and no problem in Class B can be solved in time $m^{o(k)}$. These results improve the results in [8] from two aspects: (a) the lower bounds of forms $n^{\Omega(k)}m^{O(1)}$ and $m^{\Omega(k)}$ in [8] have been improved to $f(k)n^{\Omega(k)}m^{O(1)}$ and $f(k)m^{\Omega(k)}$, respectively, for *any* function f under the same assumptions; and (b) the lower bounds of forms $n^{\Omega(k)}m^{O(1)}$ and $m^{\Omega(k)}$ in [8] were established only for *a particular* value of the parameter k, while the same lower bounds are established in the current paper for essentially *every* value of the parameter k under the same assumptions.

Note that each of the problems in Class A (respectively Class B) can be solved by a trivial algorithm of running time $cn^k m$ (respectively cm^k), where c is an absolute constant, which simply enumerates all possible subsets of k elements in the search space. Much research has tended to seek 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 [1,2,7,9,19]. Continuously improved algorithms for these problems have been developed based on improved local structure manipulations (for example, see [4,17,24,26] on the progress for the INDEPENDENT SET problem). It has even been proposed to automate the manipulation of local structures [20,25] in order to further improve the computational time.

Our results above, however, show that the power of this approach is quite limited in principle. The lower bounds $f(k)n^{\Omega(k)}p(m)$ and $f(k)m^{\Omega(k)}$ for any function f and any polynomial p mentioned above indicate that *no* local structure manipulation running in polynomial time or in time depending only on the value k will obviate the need for exhaustive enumerations.

Our techniques have also enabled us to derive lower bounds on the computational time of polynomial time approximation schemes (PTAS) for certain NP-hard problems. We pick the DISTINGUISHING SUBSTRING SELECTION problem (DSSP) as an example, for which a PTAS was recently developed [10,11]. Gramm et al. [15] showed that the parameterized DSSP problem is $W[1]$-hard, thus excluding the possibility that DSSP has a PTAS of running time $f(1/\epsilon)n^{O(1)}$ for any function f. We prove a stronger result. We first show that the DOMINATING SET problem can be linearly fpt-reduced to the DSSP problem, thus proving that the parameterized DSSP problem is $W[2]$-hard (improving the result in [15]). We then show how this lower bound on parameterized complexity can be transformed into a lower bound on the computational complexity for any PTAS for the problem. More specifically, we prove that unless all search problems in SNP are solvable in subexponential time, the DSSP problem has no PTAS of running time $f(1/\epsilon)n^{o(1/\epsilon)}$ for any function f. This essentially excludes the possibility that the DSSP problem has a practically efficient PTAS even for moderate values of the error bound ϵ. To the authors' knowledge, this is the first time a specific lower bound has been derived on the running time of a PTAS for an NP-hard problem.

We give a brief review on parameterized complexity theory. A *parameterized problem* Q is a subset of $\Omega^* \times N$, where Ω is a finite alphabet set and N is the set of all nonnegative integers. Therefore, each instance of Q is a pair (x, k), where the nonnegative integer k is called the *parameter*. The parameterized problem Q is *fixed-parameter tractable* [13] if there is an algorithm that decides whether an input (x, k) is a yes-instance of Q in time $f(k)|x|^c$, where c is a fixed constant and $f(k)$ is an arbitrary function. Denote by *FPT* the class of all fixed-parameter tractable problems.

The inherent computational difficulty for solving certain problems practically has led to the common belief that certain parameterized problems are not fixed-parameter tractable. A hierarchy of fixed-parameter intractability, *the W-hierarchy* $\bigcup_{t \geqslant 0} W[t]$, where $W[t] \subseteq W[t+1]$ for all $t \geqslant 0$, has been introduced, in which the 0th level $W[0]$ is the class *FPT*. The hardness and completeness have been defined for each level $W[i]$ of the W-hierarchy for $i \geqslant 1$ [13]. It is commonly believed that $W[1] \neq FPT$ (see [13]). Thus, $W[1]$-hardness has served as the hypothesis for fixed-parameter intractability.

In this paper, we always assume that the complexity functions in our discussions are "nice" with both domain and range being nonnegative integers and the values of the functions and their inverses can be easily computed. For two functions f and g, we write $f(n) = o(g(n))$ if there is a nondecreasing and unbounded function λ such that $f(n) \leqslant g(n)/\lambda(n)$. A function f is *subexponential* if $f(n) = 2^{o(n)}$.

2. Satisfiability and weighted satisfiability

In this section, we present two lemmas that show how a general satisfiability problem is transformed into a weighted satisfiability problem. One lemma is on circuits of bounded depth and the other lemma is on CNF formulas.

A *circuit* C of n input variables is a directed acyclic graph. The nodes of in-degree 0 are the *input gates*, each labeled uniquely either by a *positive literal* x_i or by a *negative literal* $\bar{x}_i$, $1 \leqslant i \leqslant n$. All other gates are either AND or OR gates. A special gate of out-degree 0 is designated as the *output gate*. The *size* of C is the number of gates in C, and the *depth* of C is the length of the longest path in C from an input gate to the output gate. A circuit is *monotone* (respectively *antimonotone*) if all its input gates are labeled by positive literals (respectively negative literals). A circuit represents a Boolean function in a natural way. We say that a truth assignment τ to the input variables of C *satisfies* a gate g in C if τ makes the gate g have the 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 the value 1 by τ.

A circuit C is a *Π_t-circuit* if its output gate is an AND gate and it has depth t. Using the results in [6], a Π_t-circuit C can be re-structured into an equivalent Π_t-circuit C' with size increased at most quadratically such that (1) C' has $t+1$ levels and each edge in C' only goes from a level to the next level; (2) the circuit C' has the same monotonicity and the same set of input variables; (3) level 0 of C' consists of all input gates and level t of C' consists of a single output gate; and (4) AND and OR gates in C' are organized into t alternating levels. Thus, without loss of generality, we will implicitly assume that Π_t-circuits are in this leveled form.

The SATISFIABILITY problem on Π_t-circuits, abbreviated SAT[t], is to determine if a given Π_t-circuit C has a satisfying assignment. The parameterized problem WEIGHTED SATISFIABILITY on Π_t-circuits, abbreviated WCS[t], is to determine for a given pair (C, k), where C is a Π_t-circuit and k is an integer, if C has a satisfying assignment of weight k. The WEIGHTED MONOTONE SATISFIABILITY (respectively WEIGHTED ANTIMONOTONE SATISFIABILITY) problem on Π_t-circuits, abbreviated WCS$^+[t]$ (respectively WCS$^-[t]$) is defined similarly to WCS[t] with the exception that the circuit C is required to be monotone (respectively antimonotone). It is known that for each even integer $t \geqslant 2$, WCS$^+[t]$ is $W[t]$-complete, and for each odd integer $t \geqslant 2$, WCS$^-[t]$ is $W[t]$-complete. 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.

Lemma 2.1. *Let $t \geqslant 2$ 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 $m_2 \leqslant 2m_1 + 2^{2r+1}k$, 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 $1 \leqslant i \leqslant k-1$, and $|B_k| \leqslant r$. For an integer j, $0 \leqslant j \leqslant 2^{|B_i|} - 1$, denote by $\mathrm{bin}_i(j)$ 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 hth 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 $0 \leqslant j \leqslant 2^{|B_i|} - 1$ and the hth bit in $\mathrm{bin}_i(j)$ is 1 are connected to gate g_1 in the circuit C_2. If the negative literal $\bar{x}_q$ 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 $0 \leqslant j \leqslant 2^{|B_i|} - 1$ and the hth bit in $\mathrm{bin}_i(j)$ 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 $t \geqslant 2$ 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, $1 \leqslant i \leqslant k$. Let j_i be the integer such that $\mathrm{bin}_i(j_i) = \tau_i$. 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, $1 \leqslant i \leqslant k$, 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 $\tau_i = \mathrm{bin}_i(j_i)$. 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 hth 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 $\bar{z}_{i,j}$ in block B'_i such that $0 \leqslant j \leqslant 2^{|B_i|} - 1$ and the hth bit in $\mathrm{bin}_i(j)$ is 0 are inputs to gate g_1 in C_2. If the negative literal $\bar{x}_q$ is an input to a level-1 AND gate g_2 in C_1, then all negative literals $\bar{z}_{i,j}$ in block B'_i such that $0 \leqslant j \leqslant 2^{|B_i|} - 1$ and the hth bit in $\mathrm{bin}_i(j)$ 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 $\bar{z}_{k,j}$, $2^{|B_k|} \leqslant j \leqslant s - 1$, to the output gate of the circuit C_2 (thus, the variables $z_{k,j}$, $2^{|B_k|} \leqslant j \leqslant s - 1$, 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 $\bar{z}_{i,j}$ and $\bar{z}_{i,h}$ in B'_i, $0 \leqslant j, h \leqslant 2^{|B_i|} - 1$, add an OR gate $g_{j,h}$. Connect $\bar{z}_{i,j}$ and $\bar{z}_{i,h}$ 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, $1 \leqslant i \leqslant k$. Let j_i be the integer such that $\mathrm{bin}_i(j_i) = \tau_i$. 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 $\bar{z}_{i,j_i}$ in block B'_i have the value 1, we only have to prove that no $\bar{z}_{i,j_i}$ from any block B'_i is an input to g_1. Assume to the contrary that $\bar{z}_{i,j_i}$ 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 hth variable in block B_i such that either x_q is an input to g_1 in C_1 and the hth bit of $\mathrm{bin}_i(j_i)$ is 0, or $\bar{x}_q$ is an input to g_1 in C_1 and the hth bit of $\mathrm{bin}_i(j_i)$ is 1. However, by our construction of the index j_i from the assignment τ, if the hth bit of $\mathrm{bin}_i(j_i)$ is 0 then τ assigns $x_q = 0$, and if the hth bit of $\mathrm{bin}_i(j_i)$ 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 $\bar{z}_{i,j_i}$ 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 $\bar{z}_{i,j_i}$ 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 $\bar{x}_q$ is an input to g_2 and τ assigns $x_q = 1$. Let x_q be the hth variable in block B_i. If τ assigns $x_q = 0$ then the hth 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 $\bar{z}_{i,j_i}$ would be an input to g_2 in C_2. On the other hand, if τ assigns $x_q = 1$ then the hth bit in $\text{bin}_i(j_i)$ is 1, thus, $\bar{x}_q$ cannot be an input to g_2 in C_1 because otherwise the negative literal $\bar{z}_{i,j_i}$ 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 $t \geqslant 2$, 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 $m_2 \leqslant m_1 + n_2 + k + k2^{2r} \leqslant 2m_1 + k2^{2r+1}$ (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 $O(m_2^2)$. □

Lemma 2.1 will serve as a basis for proving computational lower bounds for $W[2]$-hard problems. In order to derive similar computational lower bounds for certain $W[1]$-hard problems, we need another lemma that converts weighted satisfiability problems on monotone CNF formulas into weighted satisfiability problems on antimonotone CNF formulas.

The parameterized problem WEIGHTED MONOTONE CNF 2-SAT, abbreviated WCNF 2-SAT$^+$ (respectively WEIGHTED ANTIMONOTONE CNF 2-SAT, abbreviated WCNF 2-SAT$^-$) is: given an integer k and a CNF formula F, in which all literals are positive (respectively negative) and each clause contains at most 2 literals, determine whether there is a satisfying assignment of weight k to F.

Lemma 2.2. *There is an algorithm A_2 that, for a given integer $r > 0$, transforms each instance (F_1, k_1) of* WCNF 2-SAT$^+$, *where the formula F_1 has n_1 variables, into a group $\mathcal{G}$ of at most $(r+1)^{k_2}$ instances (F_π, k_2) of* WCNF 2-SAT$^-$, *where $k_2 = \lceil n_1/r \rceil$, and each formula F_π has $n_2 = k_2 2^r$ variables, such that (F_1, k_1) is a yes-instance of* WCNF 2-SAT$^+$ *if and only if there is a yes-instance for* WCNF 2-SAT$^-$ *in the group $\mathcal{G}$. The running time of the algorithm A_2 is bounded by $O(n_2^2(r+1)^{k_2})$.*

Proof. For the given instance (F_1, k_1) of WCNF 2-SAT$^+$, divide the n_1 variables in F_1 into $k_2 = \lceil n_1/r \rceil$ pairwise disjoint subsets $B_1, \ldots, B_{k_2}$, each containing at most r variables. Let π be a partition of the parameter k_1 into k_2 integers $h_1, \ldots, h_{k_2}$, where $0 \leqslant h_i \leqslant |B_i|$ and $k_1 = h_1 + \cdots + h_{k_2}$. We say that an assignment τ of weight k_1 for F_1 is *under the partition* π if τ assigns the value 1 to exactly h_i variables in the set B_i for every i.

Fix a partition π of the parameter k_1: $k_1 = h_1 + \cdots + h_{k_2}$. We construct an instance (F_π, k_2) for WCNF 2-SAT$^-$ as follows. For each subset $B_{i,j}$ of h_i variables in the set B_i, if for each clause (x_s, x_t) in F_1 where both x_s and x_t are in B_i, at least one of x_s and x_t is in $B_{i,j}$, then make $B_{i,j}$ a Boolean variable in F_π. Call such a $B_{i,j}$ an "essential variable" in F_π. In particular, if no clause (x_s, x_t) in F_1 has both x_s and x_t in the set B_i, then every subset of h_i variables in B_i makes an essential variable in F_π. For each pair of essential variables $B_{i,j}$ and $B_{i,q}$ in F_π from the same set B_i in F_1, add a clause $(\overline{B_{i,j}}, \overline{B_{i,q}})$ to F_π. For each pair of essential variables $B_{i,j}$ and $B_{h,q}$ in F_π from two different sets B_i and B_h in F_1, if there exist a variable $x_s \in B_i$ and a variable $x_t \in B_h$ such that $x_s \notin B_{i,j}$, $x_t \notin B_{h,q}$ but (x_s, x_t) is a clause in F_1, add a clause $(\overline{B_{i,j}}, \overline{B_{h,q}})$ to F_π. This completes the main part of the CNF formula F_π, which thus far has no more than $k_2 2^r$ variables. To make the number n_2 of variables in F_π to be exactly $k_2 2^r$, we add a proper number of "surplus" variables to F_π and for each surplus variable B' we add a unit clause $(\overline{B'})$ to F_π (so that these surplus variables are forced to have the value 0 in a satisfying assignment of F_π). Obviously, (F_π, k_2) is an instance of the WCNF 2-SAT$^-$ problem.

We verify that the CNF formula F_1 has a satisfying assignment of weight k_1 under the partition π if and only if the CNF formula F_π has a satisfying assignment of weight k_2. Let τ_1 be a satisfying assignment of weight k_1 under the partition π for F_1. Let C be the set of variables in F_1 that are assigned the value 1 by τ_1, and $C_i = C \cap B_i$. Then C_i has h_i variables. Note that for any clause (x_s, x_t) in F_1 such that both x_s and x_t are in B_i, at least one of x_s and x_t must be in C_i—otherwise the clause (x_s, x_t) would not be satisfied by the assignment τ_1. Thus, each subset C_i is an essential variable in F_π. Now in the CNF formula F_π, by assigning the value 1 to all C_i, $1 \leqslant i \leqslant k_2$, and the value 0 to all other variables (in particular, all surplus variables in F_π are assigned the value 0), we get an assignment τ_π of weight k_2 for F_π. For each clause of the form $(\overline{B_{i,j}}, \overline{B_{i,q}})$ in F_π, where $B_{i,j}$ and $B_{i,q}$ are from the same set B_i, since only one variable in F_π from the set B_i (i.e., C_i) is assigned the value 1 by τ_π, the clause is satisfied by the assignment τ_π. For two variables C_i and C_h in F_π, $i \neq h$, which both get assigned the value 1 by the assignment τ_π, each clause (x_s, x_t) in F_1 such that $x_s \in B_i$ and $x_t \in B_h$ must have either $x_s \in C_i$ or $x_t \in C_h$ (otherwise the clause (x_s, x_t) would not be satisfied by τ_1). Thus, $(\overline{C_i}, \overline{C_h})$ is not a clause in F_π. In consequence, the clauses of the form $(\overline{B_{i,j}}, \overline{B_{h,q}})$ in F_π, $i \neq h$, where $B_{i,j}$ and $B_{h,q}$ are from different sets B_i and B_h, are also all satisfied by τ_π. This shows that F_π is satisfied by the assignment τ_π of weight k_2.

Conversely, let τ_π be a satisfying assignment of weight k_2 for F_π. Because $(\overline{B_{i,j}}, \overline{B_{i,q}})$ is a clause in F_π for each pair of essential variables $B_{i,j}$ and $B_{i,q}$ from the same set B_i, at most one essential variable in F_π from each set B_i can be assigned the value 1 by the assignment τ_π. Since the weight of τ_π is k_2, we conclude that exactly one essential variable B_{i,j_i} in F_π from each set B_i is assigned the value 1 by τ_π (note that all surplus variables in F_π must be assigned the value 0 by τ_π). Each B_{i,j_i} of these subsets in F_1 contains exactly h_i variables in B_i. Let $C = \bigcup_{i=1}^{k_2} B_{i,j_i}$, then C has exactly k_1 variables in F_1. If in F_1 we assign all variables in C the value 1 and all other variables the value 0, we get an assignment τ_1 of weight k_1 for the formula F_1. We show that τ_1 is a satisfying assignment for F_1. For each clause (x_s, x_t) in F_1 where both x_s and x_t are in the same set B_i, by the construction of the essential variables in F_π, at least one of x_s and x_t is in B_{i,j_i}, and hence in C. Thus, all clauses (x_s, x_t) in F_1 where both x_s and x_t are in B_i are satisfied by the assignment τ_1. For each clause (x_s, x_t) in F_1 where $x_s \in B_i$ and $x_t \in B_h$, $i \neq h$, because $(\overline{B_{i,j_i}}, \overline{B_{h,j_h}})$ is not a clause in F_π (otherwise, τ_π would not satisfy F_π), we must have either $x_s \in B_{i,j_i}$ or $x_t \in B_{h,j_h}$, i.e., at least one of x_s and x_t must be in C. It follows that the clause (x_s, x_t) is again satisfied by τ_1. This proves that τ_1 is a satisfying assignment of weight k_1 for the formula F_1.

For each partition π of the parameter k_1, we have a corresponding instance (F_π, k_2) such that the CNF formula F_1 has a satisfying assignment of weight k_1 under the partition π if and only if (F_π, k_2) is a yes-instance of WCNF 2-SAT$^-$. Let $\mathcal{G}$ be the collection of the instances (F_π, k_2) over all partitions π of the parameter k_1. Since (F_1, k_1) is a yes-instance of WCNF 2-SAT$^+$ if and only if there is a partition π of k_1 such that F_1 has a satisfying assignment of weight k_1 under the partition π, we conclude that (F_1, k_1) is a yes-instance of WCNF 2-SAT$^+$ if and only if the group $\mathcal{G}$ contains a yes-instance of WCNF 2-SAT$^-$. The number of instances in the group $\mathcal{G}$ is bounded by the number of partitions of k_1, which is bounded by $(r+1)^{k_2}$. Finally, the instance (F_π, k_2) for a partition π of k_1 can be constructed in time $O(n_2^2)$. Therefore, the group $\mathcal{G}$ of the instances of WCNF 2-SAT$^-$ can be constructed in time $O(n_2^2(r+1)^{k_2})$. This completes the proof of the lemma. □

3. Lower bounds on weighted satisfiability problems

From Lemma 2.1, we can get a number of interesting results on the relationship between the circuit satisfiability problem SAT$[t]$ and the weighted circuit satisfiability problem WCS$^*[t]$. In the following theorems, we will denote by n the number of input variables and m the size of a circuit.

Our first result is an improvement of Theorem 3.1 in [8], where the bound $n^{o(k)}m^{O(1)}$ in [8] is improved to $f(k)n^{o(k)}m^{O(1)}$ for any nice function f (recall that a function f is *nice* if both f and the inverse of f are easily computable).

Theorem 3.1. *Let $t \geqslant 2$ 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 $f(k) \geqslant 2^k$. Define f^{-1} by $f^{-1}(h) =$

$\max\{q \mid f(q) \leqslant h\}$. Since the function f is nondecreasing and unbounded, the function f^{-1} is also nondecreasing and unbounded, and satisfies $f(f^{-1}(h)) \leqslant h$. From $f(k) \geqslant 2^k$, we have $f^{-1}(h) \leqslant \log h$.

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 2.1 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 $m_2 \leqslant 2m_1 + 2^{2r+1}k$, and the algorithm A_1 takes time $O(m_2^2)$. According to Lemma 2.1, we can determine if C_1 is a yes-instance of SAT$[t]$ by calling the algorithm $M_{\rm wcs}$ to determine if (C_2, k) is a yes-instance of WCS$^*[t]$. The running time of the algorithm $M_{\rm wcs}$ on (C_2, k) is bounded by $f(k)n_2^{k/\lambda(k)}p(m_2)$. Combining all above we get an algorithm $M_{\rm sat}$ 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 $M_{\rm sat}$ in terms of the values n_1 and m_1.

Since $k = \lceil n_1/r\rceil \leqslant f^{-1}(n_1) \leqslant \log n_1$,[6] we have $f(k) \leqslant f(f^{-1}(n_1)) \leqslant n_1$. Moreover,

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

Therefore if we set $\lambda'(n_1) = \lambda(f^{-1}(n_1)/3)$, then $\lambda(k) \geqslant \lambda'(n_1)$. 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 $k \leqslant f^{-1}(n_1) \leqslant \log n_1$),

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

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

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

Therefore, both terms $p(m_2)$ and $O(m_2^2)$ in the running time of the algorithm $M_{\rm sat}$ 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 $M_{\rm sat}$ 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. □

In fact, Theorem 3.1 remains valid even if we restrict the parameter values to be bounded by an arbitrarily small function, as shown in the following corollary.

Corollary 3.2. *Let $t \geqslant 2$ 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 $k \leqslant \mu(n)$, then the problem* SAT$[t]$ *can be solved in time $2^{o(n)}m^{O(1)}$.*

Proof. Suppose that there is an algorithm M solving the WCS$^*[t]$ problem in time $f(k)n^{o(k)}p(m)$ for parameter values $k \leqslant \mu(n)$, where p is a polynomial. Define $\mu^{-1}(h) = \max\{q \mid \mu(q) \leqslant h\}$. Since the function μ is nondecreasing and unbounded, the function μ^{-1} is also nondecreasing, unbounded, and such that $k > \mu(n)$ implies $n \leqslant \mu^{-1}(k)$.

Now we develop an algorithm that solves the WCS$^*[t]$ problem for general parameter values. For a given instance (C, k) of WCS$^*[t]$, if $k > \mu(n)$ then we enumerate all weight-k assignments to the circuit C and check if any of them satisfies the circuit, and if $k \leqslant \mu(n)$, we call the algorithm M to decide if (C, k) is a yes-instance for WCS$^*[t]$. This algorithm obviously solves the problem WCS$^*[t]$. Moreover, in case $k > \mu(n)$, the algorithm runs in time $O(2^n m^2) = O(f_1(k)m^2)$, where $f_1(k) = 2^{\mu^{-1}(k)}$, while in case $k \leqslant \mu(n)$, the algorithm runs in time $f(k)n^{o(k)}p(m)$. Therefore, the algorithm solves the problem WCS$^*[t]$ for general parameter values in time $O(f_2(k)n^{o(k)}m^{O(1)})$, where $f_2(k) = \max\{f(k), f_1(k)\}$. Now the corollary follows from Theorem 3.1. □

Further extension of the above techniques shows that similar lower bounds can be derived essentially for *every* parameter value.

---

[6] 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 $\lceil\beta\rceil \leqslant 2\beta$ and $\lfloor\beta\rfloor \geqslant \beta/2$ for any value β.

Theorem 3.3. *Let $t \geqslant 2$ be an integer and ϵ be a fixed constant, $0 < \epsilon < 1$. For any nondecreasing and unbounded function μ satisfying $\mu(n) \leqslant n^{\epsilon}$ and $\mu(2n) \leqslant 2\mu(n)$, if* WCS*$[t]$ *is solvable in time $n^{o(k)}m^{O(1)}$ for parameter values $\mu(n)/8 \leqslant k \leqslant 16\mu(n)$, then* SAT$[t]$ *is solvable in time $2^{o(n)}m^{O(1)}$.*

Proof. We first show that by properly choosing the number r in Lemma 2.1, we can make the parameter value $k = \lceil n_1/r \rceil$ satisfy the condition $\mu(n_2)/8 \leqslant k \leqslant 16\mu(n_2)$, where $n_2 = k2^r$. To show this, we extend the function μ to a continuous function by connecting $\mu(i)$ and $\mu(i+1)$ by a linear function for each integer i.

Fix the value n_1, and consider the function

$$F(z) = \mu\left(\frac{n_1 2^{z \log n_1}}{z \log n_1}\right) - \frac{n_1}{z \log n_1} = \mu\left(\frac{n_1^{z+1}}{z \log n_1}\right) - \frac{n_1}{z \log n_1}.$$

Pick a real number z_0, $0 < z_0 < 1$, such that $(z_0 \log n_1)^{1-\epsilon} \leqslant n_1^{1-(z_0+1)\epsilon}$ (for example, $z_0 = 1 - \epsilon$). For this value z_0, since $\mu(n_1^{z_0+1}/(z_0 \log n_1)) \leqslant (n_1^{z_0+1}/(z_0 \log n_1))^{\epsilon} \leqslant n_1/(z_0 \log n_1)$, we have $F(z_0) \leqslant 0$. Moreover, it is easy to check that $F(n_1/\log n_1) \geqslant 0$. Therefore, there is a real number z^* between z_0 and $n_1/\log n_1$ such that

$$\mu\left(\frac{n_1 2^{z^* \log n_1}}{z^* \log n_1}\right) \leqslant \frac{n_1}{z^* \log n_1} \quad \text{and} \quad \mu\left(\frac{n_1 2^{z^* \log n_1 + 1}}{z^* \log n_1 + 1}\right) \geqslant \frac{n_1}{z^* \log n_1 + 1}. \tag{1}$$

We explain how to find such a real number z^* efficiently. Starting from the value z_0, then the integer values $z_1 = 1$, $z_2 = 2, \ldots, \lceil n_1/\log n_1 \rceil$, we find the smallest z_i such that

$$\mu\left(\frac{n_1 2^{z_i \log n_1}}{z_i \log n_1}\right) \leqslant \frac{n_1}{z_i \log n_1} \quad \text{and} \quad \mu\left(\frac{n_1 2^{z_{i+1} \log n_1}}{z_{i+1} \log n_1}\right) \geqslant \frac{n_1}{z_{i+1} \log n_1}.$$

Now check the values $z_{i,j} = z_i + j/\log n_1$ for $j = 0, 1, \ldots, \lceil \log n_1 \rceil$ to find a j such that

$$\mu\left(\frac{n_1 2^{z_{i,j} \log n_1}}{z_{i,j} \log n_1}\right) \leqslant \frac{n_1}{z_{i,j} \log n_1} \quad \text{and} \quad \mu\left(\frac{n_1 2^{z_{i,j+1} \log n_1}}{z_{i,j+1} \log n_1}\right) \geqslant \frac{n_1}{z_{i,j+1} \log n_1}.$$

Note that $z_{i,j+1} = z_{i,j} + 1/\log n_1$ so $z_{i,j+1} \log n_1 = z_{i,j} \log n_1 + 1$. Thus, we can set $z^* = z_{i,j}$.

Now we have

$$\begin{aligned} 2\mu\left(\frac{n_1 2^{z^* \log n_1}}{z^* \log n_1}\right) &\geqslant 2\mu\left(\frac{n_1 2^{z^* \log n_1}}{z^* \log n_1 + 1}\right) \geqslant \mu\left(\frac{n_1 2^{z^* \log n_1 + 1}}{z^* \log n_1 + 1}\right) \\ &\geqslant \frac{n_1}{z^* \log n_1 + 1} \geqslant \frac{n_1}{2z^* \log n_1}, \end{aligned} \tag{2}$$

where the second inequality uses the fact $2\mu(n) \geqslant \mu(2n)$. From (1) and (2), we get

$$4\mu\left(\frac{n_1 2^{z^* \log n_1}}{z^* \log n_1}\right) \geqslant \frac{n_1}{z^* \log n_1} \geqslant \mu\left(\frac{n_1 2^{z^* \log n_1}}{z^* \log n_1}\right). \tag{3}$$

Therefore, if we set $r = \lceil z^* \log n_1 \rceil$, then from $k = \lceil n_1/r \rceil$, $n_2 = 2^r k$, and (3), we have

$$\begin{aligned} \mu(n_2) &= \mu\big(2^r k\big) = \mu\big(2^r \lceil n_1/r \rceil\big) \geqslant \mu\big(2^r n_1/r\big) \geqslant \mu\left(\frac{2^{z^* \log n_1} n_1}{2z^* \log n_1}\right) \\ &\geqslant \frac{1}{2}\mu\left(\frac{2^{z^* \log n_1} n_1}{z^* \log n_1}\right) \geqslant \frac{1}{8} \cdot \frac{n_1}{z^* \log n_1} \geqslant \frac{1}{8} \cdot \frac{n_1}{\lceil z^* \log n_1 \rceil} \\ &= \frac{1}{8} \cdot \frac{n_1}{r} \geqslant \frac{1}{16} \cdot \lceil n_1/r \rceil = \frac{k}{16}. \end{aligned}$$

On the other hand,

$$\begin{aligned} \mu(n_2) &= \mu\big(2^r k\big) \leqslant \mu\big(2^{z^* \log n_1 + 1} k\big) \leqslant 2\mu\big(2^{z^* \log n_1} \lceil n_1/r \rceil\big) \\ &\leqslant 2\mu\big(2^{z^* \log n_1 + 1} n_1/r\big) \leqslant 4\mu\left(\frac{2^{z^* \log n_1} n_1}{z^* \log n_1}\right) \leqslant \frac{4n_1}{z^* \log n_1} \\ &\leqslant \frac{8n_1}{\lceil z^* \log n_1 \rceil} = \frac{8n_1}{r} \leqslant 8\lceil n_1/r \rceil = 8k. \end{aligned}$$

This proves that the values k and n_2 satisfy the relation $\mu(n_2)/8 \leqslant k \leqslant 16\mu(n_2)$.

Now we are ready to prove our theorem. Suppose that there is an algorithm M_{wcs} of running time $n^{k/\lambda(k)}p(m)$ for the WCS*$[t]$ problem when the parameter values k are in the range $\mu(n)/8 \leqslant k \leqslant 16\mu(n)$, where $\lambda(k)$ is a nondecreasing and unbounded function and p is a polynomial. 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,
(1) let $r = \lceil z^* \log n_1 \rceil$, where z^* is the real number satisfying (1). As we explained above, the value z^* can be computed in time polynomial in n_1;
(2) call the algorithm A_1 in Lemma 2.1 on r and C_1 to construct an instance (C_2, k) of the problem WCS*$[t]$, where $k = \lceil n_1/r \rceil$, and C_2 is a Π_t-circuit of $n_2 = k2^r$ input variables and size $m_2 \leqslant 2m_1 + 2^{2r+1}k$. By the above discussion, we have $\mu(n_2)/8 \leqslant k \leqslant 16\mu(n_2)$;
(3) call the algorithm M_{wcs} on (C_2, k) to determine whether (C_2, k) is a yes-instance of WCS*$[t]$, which, by Lemma 2.1, is equivalent to whether C_1 is a yes-instance of SAT$[t]$.

The running time of steps (1) and (2) of the above algorithm is bounded by a polynomial $p_1(m_2)$ of m_2. Step (3) takes time $n_2^{k/\lambda(k)}p(m_2)$. Therefore, the total running time of this algorithm solving the SAT$[t]$ problem is bounded by $n_2^{k/\lambda(k)}p_2(m_2)$, where p_2 is a polynomial. We have (for simplicity and without affecting the correctness, we omit the floor and ceiling functions),

$$n_2^{k/\lambda(k)} = \left(2^r n_1/r\right)^{(n_1/r)/\lambda(n_1/r)} \leqslant 2^{n_1/\lambda(n_1/r)} n_1^{(n_1/r)/\lambda(n_1/r)}.$$

Now it is easy to verify that $n_2^{k/\lambda(k)} = 2^{o(n_1)}$ (observe that $k = n_1/r \geqslant \mu(n_2)/8$ hence $\lambda(n_1/r)$ is unbounded, and that $r = z^* \log n_1 = \Omega(\log n_1)$). Also, since $m_2 \leqslant 2m_1 + 2(n_2)^2$, $m_2 = 2^{o(n_1)}m_1^{O(1)}$, thus, the polynomial $p_2(m_2)$ is bounded by $2^{o(n_1)}m_1^{O(1)}$. This concludes that the above algorithm of running time $n_2^{k/\lambda(k)}p_2(m_2)$ for the problem SAT$[t]$ has its running time bounded by $2^{o(n_1)}m_1^{O(1)}$. This completes the proof of the theorem. □

Now we derive similar results for the weighted satisfiability problem WCNF 2-SAT$^-$, based on Lemma 2.2. In the following discussion, for an instance (F, k) of the problems WCNF 2-SAT$^-$ or WCNF 2-SAT$^+$, we denote by n and m, respectively, the number of variables and the instance size of the CNF formula F. Note that $m = O(n^2)$.

Theorem 3.4. *If the problem* WCNF 2-SAT$^-$ *is solvable in time* $f(k)m^{o(k)}$ (*or in time* $f(k)n^{o(k)}$) *for a function* f, *then the problem* WCNF 2-SAT$^+$ *is solvable in time* $2^{o(n)}$.

Proof. Since $m \geqslant n$ and $m = O(n^2)$ for any instance of WCNF 2-SAT$^-$, we only need to prove that if the problem WCNF 2-SAT$^-$ is solvable in time $f(k)n^{o(k)}$ for a function f, then the problem WCNF 2-SAT$^+$ is solvable in time $2^{o(n)}$.

Suppose that the problem WCNF 2-SAT$^-$ is solvable in time $f(k)n^{k/\lambda(k)}$ for a nondecreasing and unbounded function λ. Without loss of generality, we can assume that the function f is nondecreasing, unbounded, and satisfies $f(k) > 2^k$. Define $f^{-1}(h) = \max\{q \mid f(q) \leqslant h\}$. Then f^{-1} is a nondecreasing and unbounded function satisfying $f^{-1}(h) \leqslant \log h$ and $f(f^{-1}(h)) \leqslant h$.

For a given instance (F_1, k_1) of WCNF 2-SAT$^+$, where the CNF formula F_1 has n_1 variables, we let $r = \lfloor 3n_1/f^{-1}(n_1) \rfloor$ and $k_2 = \lceil n_1/r \rceil$, then we use the algorithm A_2 in Lemma 2.2 to construct a group $\mathcal{G}$ of at most $(r+1)^{k_2}$ instances (F_π, k_2) of WCNF 2-SAT$^-$, where each formula F_π has $n_2 = k_2 2^r$ variables, and such that (F_1, k_1) is a yes-instance of WCNF 2-SAT$^+$ if and only if the group $\mathcal{G}$ contains a yes-instance of WCNF 2-SAT$^-$. By our assumption, it takes time $f(k_2)n_2^{k_2/\lambda(k_2)}$ to test if each (F_π, k_2) in the group $\mathcal{G}$ is a yes-instance of WCNF 2-SAT$^-$. Therefore, in time of order

$$(r+1)^{k_2} f(k_2) n_2^{k_2/\lambda(k_2)} + n_2^2 (r+1)^{k_2},$$

we can decide if (F_1, k_1) is a yes-instance of WCNF 2-SAT$^+$, where the term $n_2^2(r+1)^{k_2}$ is for the running time of the algorithm A_2. As we verified in Theorem 3.1, $f(k_2) \leqslant n_1$, and $n_2^{k_2/\lambda(k_2)} = 2^{o(n_1)}$ (in particular, $n_2 = 2^{o(n_1)}$). Finally, since $r = O(n_1)$ and $k_2 = O(f^{-1}(n_1)) = O(\log n_1)$, we get $(r+1)^{k_2} = 2^{o(n_1)}$. In summary, in time $2^{o(n_1)}$

we can decide if (F_1, k_1) is a yes-instance of WCNF 2-SAT$^+$, and hence, the problem WCNF 2-SAT$^+$ is solvable in time $2^{o(n)}$. □

Based on Theorem 3.4, and using a proof completely similar to that of Corollary 3.2, we can prove that Theorem 3.4 remains valid even if we restrict the parameter values to be bounded by an arbitrarily small function of n.

Corollary 3.5. *Let $\mu(n)$ be any nondecreasing and unbounded function. If there is a function f such that the problem* WCNF 2-SAT$^-$ *is solvable in time $f(k)m^{o(k)}$ for parameter values $k \leqslant \mu(n)$, then the problem* WCNF 2-SAT$^+$ *is solvable in time $2^{o(n)}$.*

Theorem 3.6. *For any nondecreasing and unbounded function μ satisfying $\mu(n) \leqslant n^\epsilon$ and $\mu(2n) \leqslant 2\mu(n)$, where ϵ is a fixed constant, $0 < \epsilon < 1$, if* WCNF 2-SAT$^-$ *is solvable in time $m^{o(k)}$ (or in time $n^{o(k)}$) for parameter values $\mu(n)/8 \leqslant k \leqslant 16\mu(n)$, then the problem* WCNF 2-SAT$^+$ *is solvable in time $2^{o(n)}$.*

Proof. Again since $m = O(n^2)$, the given hypothesis implies that WCNF 2-SAT$^-$ is solvable in time $n^{o(k)}$ for parameter values $\mu(n)/8 \leqslant k \leqslant 16\mu(n)$.

Let (F_1, k_1) be an instance of WCNF 2-SAT$^+$, where the CNF formula F_1 has n_1 variables. As in Theorem 3.3, we first compute in polynomial time a real number z^* satisfying

$$4\mu\left(\frac{n_1 2^{z^* \log n_1}}{z^* \log n_1}\right) \geqslant \frac{n_1}{z^* \log n_1} \geqslant \mu\left(\frac{n_1 2^{z^* \log n_1}}{z^* \log n_1}\right).$$

Now we let $r = \lceil z^* \log n_1 \rceil$ and $k_2 = \lceil n_1/r \rceil$, and use the algorithm A_2 in Lemma 2.2 to construct a group $\mathcal{G}$ of at most $(r+1)^{k_2}$ instances (F_π, k_2) of WCNF 2-SAT$^-$, where each formula F_π has $n_2 = k_2 2^r$ variables, such that (F_1, k_1) is a yes-instance of WCNF 2-SAT$^+$ if and only if the group $\mathcal{G}$ contains a yes-instance of WCNF 2-SAT$^-$.

As proved in Theorem 3.3, the values k_2 and n_2 satisfy the relation $\mu(n_2)/8 \leqslant k_2 \leqslant 16\mu(n_2)$, and $n_2^{k_2/\lambda(k_2)} = 2^{o(n_1)}$ for any nondecreasing and unbounded function λ. Therefore, by the hypothesis of the current theorem, we can determine in time $2^{o(n_1)}$ for each (F_π, k_2) in $\mathcal{G}$ if (F_π, k_2) is a yes-instance of WCNF 2-SAT$^-$. It is also easy to verify that the total number $(r+1)^{k_2}$ of instances in the group $\mathcal{G}$ and the running time $O(n_2^2(r+1)^{k_2})$ of the algorithm A_2 are all bounded by $2^{o(n_1)}$. Therefore, using this transformation, we can determine in time $2^{o(n_1)}$ whether (F_1, k_1) is a yes-instance of WCNF 2-SAT$^+$, and hence the problem WCNF 2-SAT$^+$ is solvable in time $2^{o(n_1)}$. □

Remark. It is interesting to note, as pointed out by a anonymous referee, that the bound $n^{o(k)}m^{O(1)}$ in Theorem 3.3 and the bound $m^{o(k)}$ in Theorem 3.6, respectively, cannot be extended to $f(k)n^{o(k)}m^{O(1)}$ and $f(k)m^{o(k)}$ for an arbitrary function f. For example, consider $\mu(n) = 8\log n$. The range $\mu(n)/8 \leqslant k \leqslant 16\mu(n)$ gives $\log n \leqslant k \leqslant 128 \log n$. If we let $f(k) = 2^{k^2}$, then the brute force algorithms solve the problems WCS*$[t]$ and WCNF 2-SAT$^-$ in time $O(n^k m^2) = O(f(k)m^2)$.

4. Satisfiability problems and the W-hierarchy

The following theorem was proved in [8] (Theorem 3.2 in [8]).

Theorem 4.1. *For any integer $t \geqslant 2$, if* SAT$[t]$ *is solvable in time $2^{o(n)}m^{O(1)}$, then $W[t-1] = FPT$.*

Combining Theorem 4.1 with Theorem 3.1, Corollary 3.2, and Theorem 3.3, we get significant improvements over the results in [8].

Theorem 4.2. *For any integer $t \geqslant 2$, if the problem* WCS*$[t]$ *is solvable in time $f(k)n^{o(k)}m^{O(1)}$ for a function f, then $W[t-1] = FPT$. This theorem remains true even if we restrict the parameter values k by $k \leqslant \mu(n)$ for any nondecreasing and unbounded function μ.*

Theorem 4.3. *Let $t \geqslant 2$ be an integer and ϵ be a fixed constant, $0 < \epsilon < 1$. For any nondecreasing and unbounded function μ satisfying $\mu(n) \leqslant n^\epsilon$ and $\mu(2n) \leqslant 2\mu(n)$, if the problem* WCS*$[t]$ *is solvable in time $n^{o(k)}m^{O(1)}$ for the parameter values $\mu(n)/8 \leqslant k \leqslant 16\mu(n)$, then $W[t-1] = FPT$.*

Now we consider the satisfiability problems WCNF 2-SAT$^-$ and WCNF 2-SAT$^+$on CNF formulas. In the following discussion, for an instance (F, k) of the problems WCNF 2-SAT$^-$ or WCNF 2-SAT$^+$, we denote by n and m, respectively, the number of variables and the instance size of the formula F. Note that $m = O(n^2)$.

The class SNP introduced by Papadimitriou and Yannakakis [21] contains many well-known NP-hard problems including, for any fixed integer $q \geqslant 3$, CNF q-SAT, q-COLORABILITY, q-SET COVER, and VERTEX COVER, CLIQUE, and INDEPENDENT SET [16]. It is commonly believed that it is unlikely that all problems in SNP are solvable in subexponential time.[7] Impagliazzo, Paturi, and Zane [16] studied the class SNP and identified a group of SNP-complete problems under the SERF-reduction, in the sense that if any of these SNP-complete problems is solvable in subexponential time, then all problems in SNP are solvable in subexponential time.

Lemma 4.4. *If the problem* WCNF 2-SAT$^+$ *is solvable in time $2^{o(n)}$, then all problems in SNP are solvable in subexponential time.*

Proof. It is easy to see that the problem VERTEX COVER can be reduced to the problem WCNF 2-SAT$^+$ in a straightforward way: given an instance (G, k) of VERTEX COVER, where G is a graph of n vertices, we can construct an instance (F_G, k) of WCNF 2-SAT$^+$, where the CNF formula F_G has n variables, as follows: each vertex v_i of G makes a positive literal x_i in F_G, and each edge $[v_i, v_j]$ in G makes a clause (x_i, x_j) in F_G. It is easy to see that the graph G has a vertex cover of k vertices if and only if the CNF formula F_G has a satisfying assignment of weight k. Therefore, if the problem WCNF 2-SAT$^+$ is solvable in time $2^{o(n)}$, then the problem VERTEX COVER is solvable in subexponential time. Since VERTEX COVER is SNP-complete under the SERF-reduction [16], this in consequence implies that all problems in SNP are solvable in subexponential time. $\square$

Combining Lemma 4.4 with Theorem 3.4, Corollary 3.5, and Theorem 3.6, we get the following results.

Theorem 4.5. *If the problem* WCNF 2-SAT$^-$ *is solvable in time $f(k)m^{o(k)}$ for a function f, then all problems in SNP are solvable in subexponential time. This theorem remains true even if we restrict the parameter values k by $k \leqslant \mu(n)$ for any nondecreasing and unbounded function μ.*

Theorem 4.6. *For any nondecreasing and unbounded function μ satisfying $\mu(n) \leqslant n^\epsilon$ and $\mu(2n) \leqslant 2\mu(n)$, where ϵ is a fixed constant, $0 < \epsilon < 1$, if* WCNF 2-SAT$^-$ *is solvable in time $m^{o(k)}$ for parameter values $\mu(n)/8 \leqslant k \leqslant 16\mu(n)$, then all problems in SNP are solvable in subexponential time.*

5. Linear fpt-reductions and lower bounds

In the discussion of the problems WCS*$[t]$, we observed that besides the parameter k and the circuit size m, the number n of input variables has played an important role in the computational complexity of the problems. 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 number n of input variables in the circuits, instead of the circuit size m. Note that the circuit size m can be of the order 2^n.

Each instance (C, k) of a weighted circuit satisfiability problem such as WCS*$[t]$ can be regarded as a search problem, in which we need to select k elements from a search space consisting of a set of n input variables, and assign them the value 1 so that the circuit C is satisfied. Many well-known NP-hard problems have similar formulations. We list some of them next:

---

[7] 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* [12].

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 all clauses in F. Here the search space is the set of Boolean 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 space is $\mathcal{F}$.

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 space is U.

Many graph problems seek a subset of vertices that meet certain given conditions. For these graph problems, the natural search space is the set of all vertices. For certain problems, a polynomial time preprocessing on the input instance can significantly reduce the size of the search space. For example, for finding a vertex cover of k vertices in a graph G of n vertices, a polynomial time preprocessing can reduce the search space size to $2k$ (see [7]), based on the classical Nemhauser–Trotter theorem [18]. In the following, we present a simple algorithm for reducing the search space size for the DOMINATING SET problem (given a graph G and an integer k, decide whether there is a dominating set of k vertices, i.e., a subset D of k vertices such that every vertex not in D is adjacent to at least one vertex in D).

Suppose we are looking for a dominating set of k vertices in a graph G. Without loss of generality, we assume that G contains no isolated vertices (otherwise, we simply include the isolated vertices in the dominating set and modify the graph G and the parameter k accordingly). We say that the graph G has an *IS-Clique partition* (V_1, V_2) if the vertices of G can be partitioned into two disjoint subsets V_1 and V_2 such that V_1 makes an independent set while V_2 induces a clique. If $|V_2| \leqslant k$, then the vertices in V_2 plus any $k - |V_2|$ vertices in V_1 make a dominating set of k vertices in G. Thus, we assume that $|V_2| > k$. We claim that the graph G has a dominating set of k vertices if and only if there are k vertices in V_2 that make a dominating set for G. In fact, suppose that G has a dominating set D of k vertices, in which k_1 are in V_1 and k_2 are in V_2, where $k_1 + k_2 = k$. Now for each vertex v in $D \cap V_1$ that has no neighbor in D, we replace in D the vertex v by a neighbor u of v such that u is in V_2 (such a neighbor u must exist since V_1 is an independent set and v is not an isolated vertex). This process gives us a dominating set D' of at most k vertices in G, where D' is a subset of V_2. Adding a proper number of vertices in V_2 to D' then gives a dominating set of exact k vertices in G.

Therefore, if we are looking for a dominating set of k vertices in a graph G with an IS-Clique partition (V_1, V_2), we can restrict our search to the set of vertices in V_2, which thus makes a search space for the problem. Now we explain how to test if a given graph G has an IS-Clique partition.

Lemma 5.1. *Let the vertices of a graph G be ordered as $\{v_1, v_2, \ldots, v_n\}$ such that $\deg(v_1) \leqslant \deg(v_2) \leqslant \cdots \leqslant \deg(v_n)$ (where $\deg(v_i)$ denotes the degree of the vertex v_i). If $G = (V, E)$ has an IS-Clique partition, then either there is a vertex v_i in G where v_i and its neighbors make a clique V_2 such that $(V - V_2, V_2)$ makes an IS-Clique partition for G, or there is an index h, $1 \leqslant h \leqslant n-1$, such that $\deg(v_h) < \deg(v_{h+1})$ and $(\{v_1, \ldots, v_h\}, \{v_{h+1}, \ldots, v_n\})$ is an IS-Clique partition for G.*

Proof. Suppose that the graph G has an IS-Clique partition (V_1, V_2). We consider three different cases. (1) If there is a vertex v_i in V_2 such that v_i has no neighbor in V_1, then v_i and its neighbors make exactly the set V_2 and (V_1, V_2) is an IS-Clique partition for G; (2) If there is a vertex v_j in V_1 that is adjacent to all vertices in V_2, then v_j and its neighbors make the set $V_2 \cup \{v_j\}$, and $(V_1 - \{v_j\}, V_2 \cup \{v_j\})$ is an IS-Clique partition for G; (3) If neither of (1) and (2) is the case, then each vertex in V_2 has degree at least $|V_2|$ and each vertex in V_1 has degree at most $|V_2| - 1$. □

Using Lemma 5.1, we can develop a simple algorithm of running time $O(n^3)$ that tests if a given graph has an IS-Clique partition. Summarizing the above we obtain the following preprocessing algorithm on an instance (G, k) of the DOMINATING SET problem:

DS-Core(G, k)
(1) **if** the graph G has no IS-Clique partition,
    then let U be the entire set of vertices in G;
(2) **else** construct an IS-Clique partition (V_1, V_2) for G;
        if $|V_2| < k$,
        then let U be V_2 plus any $k - |V_2|$ vertices in V_1;
        else let $U = V_2$;
(3) return U as the search space.

The parameterized problems discussed in the current paper all share the property that they seek a subset in a search space satisfying certain properties. In most of the problems that we consider, the search space can be easily identified. For example, the search space for each of the problems WCNF-SAT, SET COVER, and HITTING SET is given as we described. For some other problems, such as DOMINATING SET, the search space can be identified by a polynomial time preprocessing algorithm (such as the **DS-Core** algorithm). If no polynomial time preprocessing algorithm is known, then we simply pick the entire input instance as the search space. For example, for the problems INDEPENDENT SET and CLIQUE, we will take the search space to be the entire vertex set. Thus, each instance of our parameterized problems is associated with a triple (k, n, m), where k is the parameter, n is the size of the search space, and m is the size of the instance. We will call such an instance a (k, n, m)-*instance*.

Theorems 4.2 and 4.5 suggest that the problem WCS*$[t]$ in the class $W[t]$ for $t \geqslant 2$ and the problem WCNF 2-SAT$^-$ in the class $W[1]$ 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. 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 of running time $f(k)n^{o(k)}m^{O(1)}$, such that on each (k, n, m)-instance x of Q, the algorithm A produces a (k', n', m')-instance x' of Q', where $k' = O(k)$, $n' = n^{O(1)}$, $m' = m^{O(1)}$, and that x is a yes-instance of Q if and only if x' is a yes-instance of Q'.

Definition. 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 $t \geqslant 2$ if the problem WCS*$[t]$ is fpt$_l$-reducible to Q_t.

Based on the above definitions and using Theorems 4.2 and 4.5, we immediately derive:

Theorem 5.2. *For $t \geqslant 2$, 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 $W[t-1] = FPT$. This remains true even if we restrict the parameter values k by $k \leqslant \mu(n)$ for any nondecreasing and unbounded function μ.*

Theorem 5.3. *No $W_l[1]$-hard parameterized problem can be solved in time $f(k)m^{o(k)}$ for a function f, unless all problems in SNP are solvable in subexponential time. This remains true even if we restrict the parameter values k by $k \leqslant \mu(n)$ for any nondecreasing and unbounded function μ.*

Using the fpt$_l$-reduction, we can immediately derive computational lower bounds for a large number of NP-hard parameterized problems.

Theorem 5.4. *The following parameterized problems are $W_l[2]$-hard*: WCNF-SAT, SET COVER, HITTING SET, *and* DOMINATING SET. *Thus, unless $W[1] = FPT$, none of them can be solved in time $f(k)n^{o(k)}m^{O(1)}$ for any function f. This theorem remains true even if we restrict the parameter values k by $k \leqslant \mu(n)$ for any nondecreasing and unbounded function μ.*

Proof. We highlight the fpt$_l$-reductions from WCS*$[2]$ = WCS$^+[2]$ to these problems, which are all we need. In fact, the reductions from WCS$^+[2]$ to the problems WCNF-SAT, HITTING SET, and SET COVER are standard and straightforward, and hence we leave them to the interested readers.

We present the fpt$_l$-reduction from WCS$^+[2]$ to DOMINATING SET here. Let (C, k) be an instance of WCS$^+[2]$, where C is a monotone Π_2-circuit. We construct a graph G_C associated with the circuit C as follows. First we remove any OR gate in C if it receives inputs from all input gates (this kind of OR gates will be satisfied by any assignment of weight larger than 0 anyway). Then we remove the output gate of C and add an edge to each pair of input gates in C. This gives the graph G_C. We claim that the circuit C has a satisfying assignment of weight k if and only if the graph G_C has a dominating set of k vertices. First observe that the graph G_C has a unique IS-Clique partition (V_1, V_2), where V_1 is the set of all OR gates and V_2 is the set of all input gates. Therefore, by the discussion before Lemma 5.1, if G_C has a dominating set D of k vertices, then we can assume that D is a subset of V_2. Now assigning the value 1 to the k input variables corresponding to the vertices in D clearly gives a satisfying assignment of weight k for the circuit C. For the other direction, from a satisfying assignment π of weight k for the circuit C, we can easily verify that the k vertices in G_C corresponding to the k input gates in C assigned the value 1 by π make a dominating set for the graph G_C. Finally, we point out that this reduction keeps the parameter value k, the search space size n (assuming that we apply the algorithm **DS-Core** to the DOMINATING SET problem), and the instance size m all unchanged. □

We remark that the reduction from WCS$^+[2]$ to DOMINATING SET presented in the proof of Theorem 5.4 also provides a new proof for the $W[2]$-hardness for the problem DOMINATING SET, which seems to be significantly simpler than the original proof given in [13].

Now we consider certain $W_l[1]$-hard problems. Define WCNF q-SAT, where $q > 0$ 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 contains at most q literals and F has a satisfying assignment of weight k.

Theorem 5.5. *The following problems are $W_l[1]$-hard*: WCNF q-SAT *for any integer $q \geqslant 2$,* CLIQUE*, and* INDEPENDENT SET. *Thus, unless all problems in SNP are solvable in subexponential time, none of them can be solved in time $f(k)m^{o(k)}$ for any function f. This theorem remains true even if we restrict the parameter values k by $k \leqslant \mu(m)$ for any nondecreasing and unbounded function μ.*

Proof. The fpt$_l$-reductions from the problem WCNF 2-SAT$^-$ to these problems are all straightforward, and hence we leave the detailed verifications to the interested readers. □

Each of the problems in Theorems 5.4 and 5.5 can be solved by a trivial 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. Much research has tended to seek 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 [1,2,7,9,19]. 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 [20,25] in order to further improve the computational time.

Theorems 5.4 and 5.5, 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 5.4 and the lower bound $f(k)m^{\Omega(k)}$ for the problems in Theorem 5.5, 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.

Weaker lower bounds, under the same assumptions in parameterized complexity theory, have been established previously [8] for the parameterized problems in Theorems 5.4 and 5.5. The main results in [8] proved that, for the case $k = \sqrt{n/\log n}$, an algorithm of running time $n^{o(k)}m^{O(1)}$ for the problems in Theorem 5.4 would imply $W[1] = FPT$, and an algorithm of running time $m^{o(k)}$ for the problems in Theorem 5.5 would imply that all problems in SNP are subexponential time solvable. However, the results in [8] do not exclude the possibility of algorithms of running time $f(k)n^{o(k)}m^{O(1)}$ for the problems in Theorem 5.4, and algorithms of running time $f(k)m^{o(k)}$ for the problems in Theorem 5.5, where f can be possibly a very large function. Moreover, the results in [8] do not claim lower bounds for the problems when the parameter value k is not equal to $\sqrt{n/\log n}$. Note that studying the complexity of NP-hard problems for parameter values other than $\sqrt{n/\log n}$, in particular for small parameter values, has been an interesting topic in research [14,22]. Moreover, after all, most research in parameterized complexity theory assumes that the parameter values are small. Therefore, Theorems 5.4 and 5.5 are very significant improvements over the results in [8].

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 Theorems 5.4 and 5.5 more efficiently. This possibility is, unfortunately, denied by the following theorems, which indicate that, essentially, the problems are actually difficult for *every* parameter value.

Theorem 5.6. *For any constant ϵ, $0 < \epsilon < 1$, and any nondecreasing and unbounded function μ satisfying $\mu(n) \leqslant n^\epsilon$, and $\mu(2n) \leqslant 2\mu(n)$, none of the problems in Theorem 5.4 can be solved in time $n^{o(k)}m^{O(1)}$ even if we restrict the parameter values k to $\mu(n)/8 \leqslant k \leqslant 16\mu(n)$, unless $W[1] = FPT$.*

Proof. As described in the proof of Theorem 5.4, each fpt$_l$-reduction from WCS$^+$[2] to a problem in Theorem 5.4 runs in time $m^{O(1)}$ and keeps the parameter value k and the search space size n unchanged. The theorem now follows directly from this fact and Theorem 4.3. □

Note that the conditions on the function μ in Theorem 5.6 are satisfied by most complexity functions, such as $\mu(n) = \log\log n$ and $\mu(n) = n^{4/5}$. Therefore, for example, unless the unlikely collapse $W[1] = FPT$ occurs, constructing a hitting set of $\log\log n$ elements requires time $n^{\Omega(\log\log n)} m^{O(1)}$, and constructing a hitting set of $\sqrt{n}$ elements requires time $n^{\Omega(\sqrt{n})} m^{O(1)}$, where n is the size of the universal set U.

Similar results hold for the problems in Theorem 5.5, by similar proofs based on Theorem 4.6.

Theorem 5.7. *For any constant ϵ, $0 < \epsilon < 1$, and any nondecreasing and unbounded function μ satisfying $\mu(n) \leqslant n^{\epsilon}$, and $\mu(2n) \leqslant 2\mu(n)$, none of the problems in Theorem 5.5 can be solved in time $m^{o(k)}$ even if we restrict the parameter values k to $\mu(n)/8 \leqslant k \leqslant 16\mu(n)$, unless all problems in SNP are subexponential time solvable.*

We observe that all problems in Theorem 5.4 are also $W_l[1]$-hard. Thus, we can actually claim stronger lower bounds for these problems in terms of the parameter value k and the instance size m, based on a stronger assumption.[8] This result will be used in the next section.

Theorem 5.8. *All problems in Theorem 5.4 are $W_l[1]$-hard. Hence, none of them can be solved in time $f(k)m^{o(k)}$ for any function f, unless all SNP problems are subexponential time solvable.*

Proof. The fpt$_l$-reduction from WCNF 2-SAT$^-$ to WCNF-SAT is straightforward. It is not difficult to verify that the fpt-reduction from WCNF-SAT to DOMINATING SET described in [13], which was originally used to prove the $W[2]$-hardness for DOMINATING SET, is actually an fpt$_l$-reduction. Finally, the fpt$_l$-reduction from DOMINATING SET to HITTING SET, and the fpt$_l$-reduction from HITTING SET to SET COVER are simple and left to the interested readers. The theorem now follows from the transitivity of the fpt$_l$-reduction, which can be easily verified. □

6. Lower bounds on approximation schemes

In this section, we discuss how the $W_l[1]$-hardness of a problem can be used to derive computational lower bounds for 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 4-tuple (I_Q, S_Q, f_Q, opt_Q), where

- I_Q is the set of input instances, which 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 $S_Q(x) = \{y \mid |y| \leqslant p(|x|) \text{ and } \pi(x, y)\}$;
- $f_Q(x, y)$ is the objective function mapping a pair $x \in I_Q$ and $y \in S_Q(x)$ to a nonnegative 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 solution $y_A(x)$ has an *approximation ratio* $r(n)$ if it satisfies the following condition:

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

[8] It can be shown that if $W[1] = FPT$ then all problems in SNP are solvable in subexponential time.

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 has an approximation ratio bounded by $r(|x|)$.

An NP optimization problem Q has a *polynomial time approximation scheme* (PTAS) if there is an algorithm A_Q that takes a pair (x, ϵ) as input, where x is an instance of Q and $\epsilon > 0$ is a real number, and returns a feasible solution y for x such that the approximation ratio of the solution y is bounded by $1 + \epsilon$, and for each fixed $\epsilon > 0$, the running time of the algorithm A_Q is bounded by a polynomial of $|x|$.

We propose the following formal framework for parameterization of NP optimization problems.

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

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

Theorem 6.1. *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 all problems in SNP are solvable in subexponential time.*

Proof. We consider the case that $Q = (I_Q, S_Q, f_Q, opt_Q)$ is a maximization problem such that the parameterized version $Q_{\geqslant}$ 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 $Q_{\geqslant}$. Consider the following algorithm $A_{\geqslant}$ for $Q_{\geqslant}$:

Algorithm $A_{\geqslant}$:

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

We verify that the algorithm $A_{\geqslant}$ solves the parameterized problem $Q_{\geqslant}$. Since Q is a maximization problem, if $f_Q(x, y) \geqslant k$ then obviously $opt_Q(x) \geqslant k$. Thus, the algorithm $A_{\geqslant}$ 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 $f_Q(x, y) \leqslant k - 1$. Since A_Q is a PTAS for Q and $\epsilon = 1/(2k)$, we must have

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

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

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

Thus, in this case the algorithm $A_{\geqslant}$ also returns a correct decision. This proves that the algorithm $A_{\geqslant}$ solves the parameterized version $Q_{\geqslant}$ of the problem Q. The running time of the algorithm $A_{\geqslant}$ 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 $Q_{\geqslant}$ is solvable in time $f(2k)m^{o(k)}$. By Theorem 5.3, all problems in SNP are solvable in subexponential time.

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

We demonstrate an application for Theorem 6.1. We pick the NP-complete 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 [11].

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, $|s_1| \leqslant |s_2|$, 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 $|s_1| \leqslant |s_2|$, 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.

DISTINGUISHING SUBSTRING SELECTION (DSSP): given a tuple (n, S_b, S_g, d_b, d_g), where n, d_b, and d_g are integers, $d_b \leqslant d_g$, $S_b = \{b_1, \dots, b_{n_b}\}$ is the set of (bad) strings, $|b_i| \geqslant n$, and $S_g = \{g_1, \dots, g_{n_g}\}$ is the set of (good) strings, $|g_j| = n$, either find a string s of length n such that $D(s, b_i) \leqslant d_b$ for all $b_i \in S_b$, and $D(s, g_j) \geqslant d_g$ for all $g_j \in S_g$, or report no such a string exists.

The DSSP problem is NP-hard [15]. Recently, Deng et al. [10] (see also [11]) developed an approximation algorithm A_d for DSSP in the following sense: for a given instance $x = (n, S_b, S_g, d_b, d_g)$ for DSSP and a real number $\epsilon > 0$, *in case x is a yes-instance*, the algorithm A_d constructs a string s of length n such that $D(s, b_i) \leqslant d_b(1+\epsilon)$ for all $b_i \in S_b$, and $D(s, g_j) \geqslant d_g(1-\epsilon)$ for all $g_j \in S_g$. The running time of the algorithm A_d is $O(m(n_b + n_g)^{O(1/\epsilon^6)})$, where m is the size of the instance. Obviously, such an algorithm is not practical even for moderate values of the error bound ϵ.

The authors of [10] called their algorithm a "PTAS" for the DSSP problem. Strictly speaking, neither the problem DSSP nor the algorithm in [10] conforms to the standard definitions of an optimization problem and a PTAS. The DSSP problem as defined above is a decision problem with no objective function specified, and it is also not clear what precise ratio the error bound ϵ measures. We will call an algorithm in the style of the one in [10] a "PTAS-[10]" for DSSP.

Since our lower bound techniques for PTAS given in Theorem 6.1 are based on the standard framework that has been widely used in the literature, we first propose an optimization version of the DSSP problem, the DSSP-OPT problem, using the standard definition of NP optimization problems. We then prove that a PTAS in the standard definition for DSSP-OPT is equivalent to a PTAS-[10] for DSSP as given in [10]. Using the systematical methods described above, we then prove that the parameterized version of DSSP-OPT is $W_l[1]$-hard, which, by Theorem 6.1, gives a computational lower bound on PTAS for DSSP-OPT. As a byproduct, this also shows that it is unlikely to have a practically efficient PTAS-[10] algorithm for the DSSP problem.

Definition. The DSSP-OPT problem is a tuple (I_D, S_D, f_D, opt_D), where

- I_D is the set of all (yes- and no-) instances in the decision version of DSSP;
- For an instance $x = (n, S_b, S_g, d_b, d_g)$ in I_D, $S_D(x)$ is the set of all strings of length n;
- For an instance $x = (n, S_b, S_g, d_b, d_g)$ in I_D and a string $s \in S_D(x)$, the objective function value $f_D(x, s)$ is defined to be the largest nonnegative integer d such that (i) $d \leqslant d_g$; (ii) $D(s, b_i) \leqslant d_b(2 - d/d_g)$ for all $b_i \in S_b$; and (iii) $D(s, g_j) \geqslant d$ 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 $d = 0, 1, \dots, d_g \leqslant n$.

We first show that a PTAS for DSSP-OPT is equivalent to a PTAS-[10] for DSSP. Since the PTAS-[10] for DSSP is only for yes-instances of DSSP, we will concentrate on the performance of the algorithms for yes-instances of the problem DSSP.

Lemma 6.2. *The* DSSP-OPT *problem has a PTAS of running time $\phi(m, 1/\epsilon)$ if and only if there is an algorithm A_d of running time $\phi(m, O(1/\epsilon))$ for* DSSP *that for any yes-instance of* DSSP (n, S_b, S_g, d_b, d_g) *and $\epsilon > 0$, constructs a string s of length n such that $D(s, b_i) \leqslant d_b(1+\epsilon)$ for all $b_i \in S_b$, and $D(s, g_j) \geqslant d_g(1-\epsilon)$ for all $g_j \in S_g$.*

Proof. Since $x = (n, S_b, S_g, d_b, d_g)$ is assumed to be a yes-instance of the decision problem DSSP, when x is regarded as an instance for the optimization problem DSSP-OPT, we have $opt_D(x) = d_g$.

Suppose the DSSP-OPT problem has a PTAS A_p of running time $\phi(m, 1/\epsilon)$. We show for a yes-instance $x = (n, S_b, S_g, d_b, d_g)$ and $\epsilon > 0$ how to construct a string s such that $D(s, b_i) \leqslant d_b(1+\epsilon)$ for all $b_i \in S_b$, and $D(s, g_j) \geqslant d_g(1-\epsilon)$ for all $g_j \in S_g$. Let $\epsilon' = \epsilon/(1-\epsilon)$ (note that $1/\epsilon' = O(1/\epsilon)$). Apply the PTAS A_p on x and ϵ', we get a string s_p of length n such that $f_D(x, s_p) = d_p$, $opt_D(x)/d_p = d_g/d_p \leqslant 1 + \epsilon'$, and

$$D(s_p, b_i) \leqslant d_b(2 - d_p/d_g) \quad \text{for all } b_i \in S_b,$$

and

$$D(s_p, g_j) \geqslant d_p \quad \text{for all } g_j \in S_g.$$

Now from $d_p \geqslant d_g/(1+\epsilon') = d_g(1-\epsilon)$, we get $D(s_p, g_j) \geqslant d_g(1-\epsilon)$ for all $g_j \in S_g$. From

$$2 - d_p/d_g \leqslant 2 - 1/(1+\epsilon') = 1+\epsilon,$$

we get $D(s_p, b_i) \leqslant d_b(1+\epsilon)$ for all $b_i \in S_b$. The running time of the algorithm A_p is $\phi(m, 1/\epsilon') = \phi(m, O(1/\epsilon))$. This shows that a PTAS-[10] of running time $\phi(m, O(1/\epsilon))$ for DSSP can be constructed based on the PTAS A_p for the DSSP-OPT problem.

Conversely, suppose that we have a PTAS-[10] A_d of running time $\phi(m, 1/\epsilon)$ for DSSP. We show how to construct a PTAS for the DSSP-OPT problem. For an instance $x = (n, S_b, S_g, d_b, d_g)$ of DSSP-OPT and $\epsilon > 0$, we call the algorithm A_d on x and $\epsilon' = \epsilon/(2+2\epsilon)$. By our assumption, if x is a yes-instance, then the algorithm A_d returns a string s_d of length n such that $D(s_d, b_i) \leqslant d_b(1+\epsilon')$ for all $b_i \in S_b$, and $D(s_d, g_j) \geqslant d_g(1-\epsilon')$ for all $g_j \in S_g$. We first consider the value $f_D(x, s_d)$ for DSSP-OPT. Let $d = d_g - \lceil \epsilon' d_g \rceil$. Then for each good string g_j, we have

$$D(s_d, g_j) \geqslant d_g(1-\epsilon') = d_g - \epsilon' d_g \geqslant d_g - \lceil \epsilon' d_g \rceil = d,$$

and since $d = d_g - \lceil \epsilon' d_g \rceil \leqslant d_g - \epsilon' d_g = d_g(1-\epsilon')$, for each bad string b_i,

$$D(s_d, b_i) \leqslant d_b(1+\epsilon') = d_b\big(2 - (1-\epsilon')\big) \leqslant d_b(2 - d/d_g).$$

By the definition of the function $f_D(x, s_d)$, we have $f_D(x, s_d) \geqslant d = d_g - \lceil \epsilon' d_g \rceil$.

Now consider the ratio $opt_D(x)/f_D(x, s_d)$ for the string s_d. If $\epsilon' d_g < 0.5$, then (note that $d_b \leqslant d_g$)

$$D(s_d, b_i) \leqslant d_b(1+\epsilon') < d_b + 0.5 \quad \text{and} \quad D(s_d, g_j) \geqslant d_g(1-\epsilon') > d_g - 0.5.$$

Since all $D(s_d, b_i)$, d_b, $D(s_d, g_j)$, and d_g are integers, we have $D(s_d, b_i) \leqslant d_b = d_b(2 - d_g/d_g)$ for all $b_i \in S_b$, and $D(s_d, g_j) \geqslant d_g$ for all $g_j \in S_g$. Therefore, we have $f_D(x, s_d) = d_g$ and $opt(x)/f_D(x, s_d) = 1$. On the other hand, if $\epsilon' d_g \geqslant 0.5$, then $d_g - \lceil \epsilon' d_g \rceil \geqslant d_g - 2\epsilon' d_g$, and we have

$$\begin{aligned} opt(x)/f_D(x, s_d) &\leqslant d_g/\big(d_g - \lceil \epsilon' d_g \rceil\big) \leqslant d_g/(d_g - 2\epsilon' d_g) \\ &= 1/(1-2\epsilon') = 1+\epsilon. \end{aligned}$$

Therefore, in all cases, the string s_d produced by the algorithm A_d is a solution of approximation ratio $1+\epsilon$ for the instance x of DSSP-OPT. Again, the running time of the algorithm is dominated by that of A_d, which is bounded by $\phi(m, 1/\epsilon') = \phi(m, O(1/\epsilon))$.

This completes the proof of the lemma. □

Lemma 6.2 shows that a PTAS-[10] for the problem DSSP is also a PTAS in the standard definition for the optimization problem DSSP-OPT.

Now using the standard parameterization of optimization problems, we can study the parameterized complexity of the problem DSSP-OPT$_\geqslant$.

Lemma 6.3. *The parameterized problem* DSSP-OPT$_\geqslant$ *is* $W_l[1]$*-hard.*

Proof. We prove the lemma by an fpt$_l$-reduction from the $W_l[1]$-hard problem DOMINATING SET to the DSSP-OPT$_\geqslant$ problem (see Theorem 5.8).

Let (G, k) be an instance of the DOMINATING SET problem. Suppose that the graph G has n vertices $v_1, \ldots, v_n$. Denote by $\mathrm{vec}(v_i)$ the binary string of length n in which all bits are 0 except the ith bit is 1. The instance $x_G = (n', S_b, S_g, d_b, d_g)$ for DSSP-OPT is constructed as follows: $n' = n + 5$, S_g consists of a single string $g_0 = 0^{n+5}$, $d_b = k - 1$, and $d_g = k + 3$.

The bad string set $S_b = \{b_1, \ldots, b_n\}$ consists of n strings, where b_i corresponds to the vertex v_i in G. Suppose the neighbors of the vertex v_i in G are $v_{i_1}, \ldots, v_{i_r}$, then the string b_i takes the form

$$\begin{aligned} &\mathrm{vec}(v_i) \cdot 02220 \cdot \mathrm{vec}(v_i) \cdot 00000 \cdot \mathrm{vec}(v_{i_1}) \cdot 02220 \cdot \mathrm{vec}(v_{i_1}) \\ &\quad \cdot 00000 \cdot \cdots \cdot 00000 \cdot \mathrm{vec}(v_{i_r}) \cdot 02220 \cdot \mathrm{vec}(v_{i_r}), \end{aligned}$$

where the dots "·" stand for string concatenations. It is easy to see that the size of x_G is bounded by a polynomial of the size of the graph G. Finally, we set the parameter $k' = k + 3$. Thus, (x_G, k') makes an instance for the DSSP-OPT$_{\geqslant}$ problem.

We prove that (G, k) is a yes-instance for DOMINATING SET if and only if (x_G, k') is a yes-instance for DSSP-OPT$_{\geqslant}$. Suppose the graph G has a dominating set H of k vertices. Let vec(H) be the binary string of length n whose hth bit is 1 if and only if $v_h \in H$. Now consider the string $s = \mathrm{vec}(H) \cdot 02220$. Clearly $D(s, g_0) = k + 3 = d_g$. For each bad string b_i, since H is a dominating set, either $v_i \in H$ or a vertex $v_j \in H$ is a neighbor of v_i. If $v_i \in H$ then the substring $b_i' = \mathrm{vec}(v_i) \cdot 02220$ in b_i satisfies $D(s, b_i') = k - 1$, and if a vertex $v_j \in H$ is a neighbor of v_i, then the substring $b_i' = \mathrm{vec}(v_j) \cdot 02220$ in b_i satisfies $D(s, b_i') = k - 1$. This verifies that $D(s, b_i) = k - 1 = d_b(2 - d_g/d_g)$ for all $1 \leqslant i \leqslant n$. Thus, for the string s, we have $f_D(x_G, s) = opt_D(x_G) = d_g = k + 3 \geqslant k'$. In consequence, (x_G, k') is a yes-instance of DSSP-OPT$_{\geqslant}$.

Conversely, suppose (x_G, k') is a yes-instance for the DSSP-OPT$_{\geqslant}$ problem. Then there is a string s of length $n + 5$ such that $f_D(x_G, s) = d \geqslant k' = k + 3$. By the definition, $f_D(x_G, s) \leqslant d_g = k + 3$. Thus, we must have $d = k + 3$. From the definition of the integer d, we have $D(s, g_0) \geqslant d = k + 3$, and $D(s, b_i) \leqslant d_b(2 - d/d_g) = d_b = k - 1$ for all bad strings b_i. Since $g_0 = 0^{n+5}$ and $D(s, g_0) \geqslant k + 3$, s has at least $k + 3$ "non-0" bits. On the other hand, it is easy to see that each substring of length $n + 5$ in any bad string b_i contains at most 4 "non-0" bits. Since $D(s, b_i) \leqslant k - 1$ for each bad string b_i, the string s should not contain more than $k + 3$ "non-0" bits. Thus, the string s has exactly $k + 3$ "non-0" bits. Now consider any substring b_i' of length $n + 5$ in a bad string b_i such that $D(s, b_i') \leqslant k - 1$. The substring b_i' must contain "222": otherwise b_i' has at most three "non-0" bits so $D(s, b_i') \leqslant k - 1$ would not be possible. If the substring "222" in b_i' does not match three "2"'s in s, then s has at least k "non-0" bits in other places while b_i' has only one "non-0" bit in other place, so $D(s, b_i') \leqslant k - 1$ would not be possible. Thus, the string s must contain the substring "222," which matches the substring "222" in b_i'. Finally, observe that we can always assume that the string s ends with "02220"—otherwise we simply cyclically shift the string s to move the substring "02220" to the end. Note if $D(s, b_i') \leqslant k - 1$ and b_i' is a substring in a segment "$00000 \cdot \mathrm{vec}(v_j) \cdot 02220 \cdot \mathrm{vec}(v_j) \cdot 00000$" in the bad string b_i, then after shifting s, we must have $D(s, b_i'') \leqslant k - 1$, where $b_i'' = \mathrm{vec}(v_j) \cdot 02220$. Therefore, if s is a solution to the instance (x_G, k'), then so is the string after the cyclic shifting.

Thus, the string s can be assumed to have the form $s' \cdot 02220$, where s' is a string of length n, with exactly k "non-0" bits. Suppose that the j_1th, j_2th, ..., and j_kth bits of s' are "non-0." We claim that the vertex set $H_s = \{v_{j_1}, \ldots, v_{j_k}\}$ makes a dominating set of k vertices for the graph G. In fact, for any bad string b_i, let b_i' be a substring of length $n + 5$ in b_i such that $D(s, b_i') \leqslant k - 1$. According to the above discussion, b_i' must be of the form $\mathrm{vec}(v_j) \cdot 02220$, where either $v_j = v_i$ or v_j is a neighbor of v_i. The only "non-0" bit in vec(v_j) is the jth bit, and j must be among $\{j_1, \ldots, j_k\}$—otherwise $D(\mathrm{vec}(v_j), s')$ is at least $k + 1$. Therefore, if $v_i = v_j$ then $v_i \in H_s$, and if v_j is a neighbor of v_i, then v_i is adjacent to the vertex v_j in H_s. This proves that H_s is a dominating set of k vertices in G, and that (G, k) is a yes-instance for DOMINATING SET.

This completes the proof that the problem DOMINATING SET is fpt$_l$-reducible to the problem DSSP-OPT$_{\geqslant}$. In consequence, DSSP-OPT$_{\geqslant}$ is $W_l[1]$-hard. □

We remark that the problem DOMINATING SET is $W[2]$-hard under the regular fpt-reduction [13]. Therefore, the proof of Lemma 6.3 actually shows that the DSSP-OPT$_{\geqslant}$ problem is $W[2]$-hard. This improves the result in [15], which proved that the problem is $W[1]$-hard.

From Lemma 6.3 and Theorem 6.1, we get the following result.

Theorem 6.4. *Unless all SNP problems are solvable in subexponential time, the optimization problem* DSSP-OPT *has no PTAS of running time $f(1/\epsilon)m^{o(1/\epsilon)}$ for any function f.*

By Lemma 6.2, a PTAS-[10] of running time $f(1/\epsilon)m^{o(1/\epsilon)}$ for DSSP would imply a PTAS of running time $f'(1/\epsilon)m^{o(1/\epsilon)}$ for DSSP-OPT for a function f'. Therefore, Theorem 6.4 also implies that any PTAS-[10] for DSSP cannot run in time $f(1/\epsilon)m^{o(1/\epsilon)}$ for any function f. Thus essentially, no PTAS-[10] for DSSP can be practically efficient even for moderate values of the error bound ϵ. To the authors' knowledge, this is the first time a specific lower bound is derived on the running time of a PTAS for an NP-hard problem.

Theorem 6.4 also demonstrates the usefulness of our techniques. 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 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-OPT 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-OPT has PTAS. Thus, from the viewpoint of approximability, DOMINATING SET is much harder than DSSP-OPT, and our linear fpt-reduction reduces a harder problem to an easier problem. This hints that our approach for deriving computational lower bounds *cannot* be simply replaced by the standard approaches based on approximation ratio-preserving reductions.

7. Conclusion

In this paper, based on parameterized complexity theory, we developed new techniques for deriving computational lower bounds for well-known NP-hard problems. We started by establishing the computational lower bounds for the generic parameterized problems WCS*$[t]$ for $t \geqslant 2$ and WCNF 2-SAT$^-$. We showed that for any integer $t \geqslant 2$, an $f(k)n^{o(k)}m^{O(1)}$-time algorithm for WCS*$[t]$ for any function f would collapse the $(t-1)$st level $W[t-1]$ to the bottom level *FPT* in the fixed-parameter intractability hierarchy, the W-hierarchy, and that an $f(k)m^{o(k)}$-time algorithm for WCNF 2-SAT$^-$ would imply subexponential time algorithms for all problems in SNP. Based on these generic results, we introduced the concept of linear fpt-reductions, and used it to derive tight computational lower bounds for many well-known NP-hard problems. Obviously, the list of the problems we have given here is far from being exhaustive. This new technique should serve as a very powerful tool for deriving strong computational lower bounds for other intractable problems. Moreover, we demonstrated how our techniques can be used to derive strong computational lower bounds on polynomial time approximation schemes for NP-hard problems. This seems to open a new direction for the study of computational lower bounds on the approximability of NP-hard optimization problems.

Acknowledgments

The authors acknowledge the helpful and stimulating discussions with Liming Cai, Mike Fellows, Anna Gal, Martin Grohe, David Juedes, Janos Pach, and Vijaya Ramachandran. The authors specially thank two anonymous referees, whose comments and corrections have improved the presentation of the paper. In particular, one of the referees made a nice observation that the bounds in Theorems 3.3 and 3.6 cannot be further improved.

References

[1] K.A. Abrahamson, R.G. Downey, M.R. Fellows, Fixed-parameter tractability and completeness IV: On completeness for $W[P]$ and PSPACE analogs, Ann. Pure Appl. Logic 73 (1995) 235–276.
[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 (2002) 461–493.
[3] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, M. Protasi, Complexity and Approximation – Combinatorial Optimization Problems and Their Approximability Properties, Springer-Verlag, 1999.
[4] R. Beigel, Finding maximum independent sets in sparse and general graphs, in: Proc. 10th Annual ACM–SIAM Symp. on Discrete Algorithms, SODA '99, 1999 pp. 856-857.
[5] J. Cheetham, F. Dehne, A. Rau-Chaplin, U. Stege, P. Taillon, Solving large FPT problems on coarse grained parallel machines, J. Comput. System Sci. 67 (2003) 691–701.
[6] J. Chen, Characterizing parallel hierarchy by reducibilities, Inform. Process. Lett. 39 (1991) 303–307.
[7] J. Chen, I.A. Kanj, W. Jia, Vertex cover: Further observations and further improvements, J. Algorithms 41 (2001) 280–301.
[8] J. Chen, B. Chor, M. Fellows, X. Huang, D. Juedes, I. Kanj, G. Xia, Tight lower bounds for certain parameterized NP-hard problems, Inform. and Comput. 201 (2005) 216–231.
[9] D. Coppersmith, S. Winograd, Matrix multiplication via arithmetic progression, J. Symbolic Comput. 9 (1990) 251–280.
[10] X. Deng, G. Li, Z. Li, B. Ma, L. Wang, A PTAS for distinguishing (sub)string selection, in: ICALP '02, in: Lecture Notes in Comput. Sci., vol. 2380, 2002, pp. 740–751.
[11] X. Deng, G. Li, Z. Li, B. Ma, L. Wang, Genetic design of drugs without side-effects, SIAM J. Comput. 32 (2003) 1073–1090.
[12] R. Downey, V. Estivill-Castro, M. Fellows, E. Prieto-Rodriguez, F. Rosamond, Cutting up is hard to do: The parameterized complexity of k-cut and related problems, Electron. Notes Theor. Comput. Sci. 78 (2003) 205–218.
[13] R.G. Downey, M.R. Fellows, Parameterized Complexity, Springer-Verlag, 1999.
[14] U. Feige, J. Kilian, On limited versus polynomial nondeterminism, Chic. J. Theoret. Comput. Sci. 1997 (1997).

[15] J. Gramm, J. Guo, R. Niedermeier, On exact and approximation algorithms for distinguishing substring selection, in: FCT '2003, in: Lecture Notes in Comput. Sci., vol. 2751, 2003, pp. 195–209.
[16] R. Impagliazzo, R. Paturi, F. Zane, Which problems have strongly exponential complexity?, J. Comput. System Sci. 63 (2001) 512–530.
[17] T. Jian, An $O(2^{0.304n})$ algorithm for solving maximum independent set problem, IEEE Trans. Comput. 35 (1986) 847–851.
[18] G. Nemhauser, L. Trotter, Vertex packing: Structural properties and algorithms, Math. Program. 8 (1975) 232–248.
[19] J. Nešetřil, S. Poljak, On the complexity of the subgraph problem, Comment. Math. Univ. Carolin. 26 (1985) 415–419.
[20] R. Niedermeier, P. Rossmanith, Upper bounds for vertex cover further improved, in: STACS '99, in: Lecture Notes in Comput. Sci., vol. 1563, 1999, pp. 561–570.
[21] C.H. Papadimitriou, M. Yannakakis, Optimization, approximation, and complexity classes, J. Comput. System Sci. 43 (1991) 425–440.
[22] C.H. Papadimitriou, M. Yannakakis, On limited nondeterminism and the complexity of VC dimension, J. Comput. System Sci. 53 (1996) 161–170.
[23] C.H. Papadimitriou, M. Yannakakis, On the complexity of database queries, J. Comput. System Sci. 58 (1999) 407–427.
[24] J.M. Robson, Algorithms for maximum independent sets, J. Algorithms 7 (1986) 425–440.
[25] J.M. Robson, Finding a maximum independent set in time $O(2^{n/4})$?, LaBRI, Université Bordeaux I, 1251-01, 2001.
[26] R.E. Tarjan, A.E. Trojanowski, Finding a maximum independent set, SIAM J. Comput. 6 (1977) 537–546.