ACADEMIC PRESS

Available at www.ComputerScienceWeb.com POWERED BY SCIENCE DIRECT®

Journal of Computer and System Sciences 67 (2003) 789–807

JOURNAL OF COMPUTER AND SYSTEM SCIENCES

http://www.elsevier.com/locate/jcss

On the existence of subexponential parameterized algorithms ☆

Liming Cai[a] and David Juedes[b,*]

[a] *Department of Computer Science, University of Georgia, Athens, GA 30602, USA*

[b] *School of Electrical Engineering and Computer Science, Ohio University, Athens, OH 45701, USA*

Received 1 October 2001; revised 2 May 2002

Abstract

The existence of subexponential-time parameterized algorithms is examined for various parameterized problems solvable in time $O(2^{O(k)}p(n))$. It is shown that for each $t \geqslant 1$, there are parameterized problems in FPT for which the existence of $O(2^{o(k)}p(n))$-time parameterized algorithms implies the collapse of $W[t]$ to FPT. Evidence is demonstrated that Max-SNP-hard optimization problems do not admit subexponential-time parameterized algorithms. In particular, it is shown that each Max-SNP-complete problem is solvable in time $O(2^{o(k)}p(n))$ if and only if 3-SAT $\in$ DTIME$(2^{o(n)})$. These results are also applied to show evidence for the non-existence of $O(2^{o(\sqrt{k})}p(n))$-time parameterized algorithms for a number of other important problems such as Dominating Set, Vertex Cover, and Independent Set on planar graph instances.

1. Introduction

Recent substantial progress has been made in building better and better parameterized algorithms for a variety of NP-complete problems. Consider the problem of determining whether a graph with n nodes has a Vertex Cover of size k. Starting with the early work of Buss [7,8] who discovered a $O(2^k k^{2k+2} + kn)$ algorithm for the problem, the running time of parameterized algorithms for Vertex Cover has been improved to $O(2^k k^2 + kn)$ by Downey and Fellows [15], $O(1.325^k k^2 + kn)$ by Balasubramanian et al. [5], $O(1.3196^k k^2 + kn)$ by Downey et al. [17], $O(1.29175^k k^2 + kn)$ and $O(1.29175^k + kn)$ by Niedermeier and Rossmanith [22,23], and $O(1.2852^k + kn)$ by Chen et al. [13]. Similar improvements have been made for other NP-complete problems [4]. In particular, we mention the case for Planar Dominating Set. As shown

☆ This work was supported in part by National Science Foundation research grant CCR-000246.

* Corresponding author.

E-mail addresses: cai@cs.uga.edu (L. Cai), juedes@ohiou.edu (D. Juedes).

0022-0000/03/$ - see front matter
doi:10.1016/S0022-0000(03)00074-6

by Downey and Fellows [15], this problem is known to be fixed parameter tractable via a $O(11^k|G|)$ algorithm. However, this result was recently improved to $O(2^{O(\sqrt{k})}n)$ by Alber et al. [2].

Noting the progress on algorithms for Planar Dominating Set, it is natural to ask if similar progress can be made for other problems such as Vertex Cover. In particular, it is natural to ask whether the current $O(2^{O(k)}p(n))$ upper bound on Vertex Cover can be improved to $O(2^{o(k)}p(n))$. This is an open question.

In this paper, we examine the complexity theoretic consequences of the existence of subexponential parameterized algorithms for a variety of parameterized problems, including Vertex Cover. In the case of Vertex Cover, we relate the existence of subexponential-time parameterized algorithms to the existence of subexponential-time algorithms for 3-SAT. For other problems, we show that the existence of the subexponential parameterized algorithms implies the collapse of the W-hierarchy. Our approach in the second case is based on generalized parameterizations of optimization problems.

For each function $s(n)$ and each parameterized problem Π with parameter k, we define the *extended version* $\Pi^{s(n)}$ to be the problem Π with the role of parameter k replaced by $ks(n)$. We show that the parameterized complexity of $\Pi^{s(n)}$ depends largely on the function s as well as that of Π. In particular, there are problems $\Pi \in \mathrm{FPT}$ whose extended versions $\Pi^{s(n)}$ remain parameterized tractable when $s = o(\log n)$ but become parameterized intractable when $s = \Theta(\log n)$. We are able to show that for each $t \geqslant 1$, there is a parameterized problem Π that is solvable in time $O(2^{O(k)}p(n))$ whose extended version $\Pi^{\log n}$ is hard for $W[t]$. Our techniques allow us to show that the existence of $O(2^{o(k)}p(n))$-time parameterized algorithms for Π would imply parameterized tractability for $\Pi^{\log n}$ causing the collapse of $W[t]$ to FPT. Our results are of special interest when $t = 1$. In this case, the problem Π of interest is a weighted satisfiability problem Half Weight SAT. The problem contains Vertex Cover as a special case and yet itself is a special case of the general SAT problem.

Using the same techniques, an earlier version of this work [12] concluded that Vertex Cover and other Max-SNP-hard problems do not have subexponential parameterized algorithms unless $W[1] = \mathrm{FPT}$. However, a careful examination[1] revealed that the proof of Lemma 3 in [12] was incorrect. Since subsequent results ([12, Theorem 4, Theorem 6, Corollary 3]) relied on Lemma 3, the conclusion that Vertex Cover does not have a subexponential parameterized algorithm unless $W[1] = \mathrm{FPT}$ remains open. Nevertheless, the results presented here and in [12] provide a blueprint for parameterized lower-bound proofs. In the case of Vertex Cover, it suffices to prove that Vertex Cover$^{\log n}$ is $W[1]$-hard. By Theorem 2.2 of this paper, the existence of a subexponential-time parameterized algorithm for Vertex Cover would imply that Vertex Cover$^{\log n}$ is parameterized tractable and hence that $W[1] = \mathrm{FPT}$. So, it appears that an investigation of the parameterized complexity of Vertex Cover$^{\log n}$ and other related problems may be worthwhile. In a recent development, Fellows [18] has shown that Vertex Cover$^{\log n}$ is contained in $W[1]$.

Here we examine the parameterized complexity of Vertex Cover and other Max-SNP-hard problems. It is known from the earlier work of Cai and Chen [9] that under the standard parameterization framework, all parameterized versions of the optimization problems in the class

[1] During the workshop on parameterized complexity at Schloss Dagstuhl in the summer of 2001.

Max-SNP are solvable in time $O(2^{O(k)}p(n))$. This research gives a more detailed account on the parameterized complexity of Max-SNP. We show that L-reductions preserve subexponential-time parameterized computability and that no Max-SNP-complete problem is solvable in time $O(2^{o(k)}p(n))$ unless all problems in Max-SNP are solvable in this time-bound. Moreover, we relate the existence of subexponential-time algorithms for Max-SNP problems to the long-standing question of whether 3-SAT is solvable in a subexponential time.

Our results concerning the parameterized complexity of Max-SNP-complete problems use the work of Impagliazzo et al. [20] in combination with reduction to kernal techniques. Their work indicates that Vertex Cover and other NP-complete problems likely do not have subexponential parameterized algorithms. In particular, their work defines a notion of completeness under "SERF" (subexponential reduction family) reductions for the syntactic class SNP that was originally defined by Papadimitriou and Yannakakis [24]. As shown there, the existence of a subexponential-time algorithm for any problem that is SNP-hard under SERF reductions implies that every problem in SNP has a subexponential-time algorithm. In their work, many NP-complete problems, including Vertex Cover, Independent Set, and 3-SAT, where shown to be SNP-hard under SERF reductions. In the case of Vertex Cover, since k is bounded above by n, the existence of a subexponential-time parameterized algorithm for Vertex Cover implies the existence of a subexponential-time algorithm for the usual decision version. Therefore, a subexponential-time parameterized algorithm for Vertex Cover implies that 3-SAT has a subexponential-time algorithm.

Our results are extended to obtain parameterized complexity lower bounds for a number of problems on instances with planar structures. We show evidence that for a number of problems on planar graphs, the $O(2^{O(\sqrt{k})}p(n))$-time algorithms obtained by Alber et al. [2] cannot be substantially improved in the exponent $O(\sqrt{k})$. In particular, we show that problems such as Planar Dominating Set, Planar Independent Set, Planar Red/Blue Dominating Set, and Planar Vertex Cover cannot be solved in time $O(2^{o(\sqrt{k})}p(n))$ unless all Max-SNP-complete problems are solvable in time $O(2^{o(k)}p(n))$. Putting this along with the results established for Max-SNP-complete problems, we show that the existence of $O(2^{o(\sqrt{k})}p(n))$-time algorithms for problems on planar structures also relates to the long-standing question of whether 3-SAT $\in$ DTIME$(2^{o(n)})$.

1.1. Outline of main results

The paper is organized as follows. Section 2 introduces the extended parameterization framework. The main result of Section 2 is Theorem 2.2, i.e., that if a parameterized problem Π is solvable in time $O(2^{o(k)}p(n))$ for some polynomial p, then $\Pi^{\log n}$ is parameterized tractable. Section 3 studies the relationship between the hardness of $W[t]$-complete problems and the existence of subexponential-time algorithms for their variants. In particular, Section 3 introduces a family of parameterized problems Half-Weight t-Norm Sat. In Lemma 3.1, it is shown that the extended versions of these parameterized problems, Half-Weight $(t+1)$-Norm Sat$^{\log n}$, are $W[t]$-hard via parameterized reductions from the $W[t]$-complete problems Weighted Monotone t-Normalized Satisfiability and Weighted Antimonotone t-Normalized Satisfiability [16]. Using Theorem 2.2, we

conclude that Half-Weight $t+1$-Norm Sat cannot be solved in $O(2^{o(k)}p(n))$ steps unless $W[t] =$ FPT.

In Section 4, we examine the complexity theoretic consequences of subexponential-time parameterized algorithms for Vertex Cover and other Max-SNP-hard problems. We do so by relating the complexity of Vertex Cover-3 with the complexity of 3-SAT. There we use the results of Impagliazzo et al. [20, Theorem 1, Section 3] that show that Vertex Cover-3 with the parameter n (the number of vertices), 3-SAT with the parameter n (the number of variables), and 3-SAT with the parameter m (the number of clauses) are also SNP-complete under SERF reductions. By using these results in combination with a reduction to kernal, we show in Theorem 4.3 that Vertex Cover-3 can be solved in $O(2^{o(k)}p(n))$ steps if and only if 3-SAT is in DTIME$(2^{o(n)})$, where n is the number of variables. It follows in Theorem 4.4 that each Max-SNP-complete problem Π can be solved in time $O(2^{o(k)}p(n))$ with witness for some polynomial p if and only if 3-SAT $\in$ DTIME$(2^{o(n)})$, where n is the number of variables.

In Section 5, we examine the complexity theoretic consequences of the existence of $O(2^{o(\sqrt{k})}p(n))$ parameterized algorithms for planar problems. We do so by relating the parameterized complexity of Planar Vertex Cover, Planar Independent Set, Planar Dominating Set, and Planar Red/Blue Dominating Set to the parameterized complexity of Vertex Cover-3. In particular, we use a reduction to kernal, along with an earlier NP-completeness reduction from Planar Vertex Cover to Vertex Cover by Garey et al. [19] to show that the existence of a $O(2^{o(\sqrt{k})}p(n))$ algorithm for any of these problems implies the existence of a $O(2^{o(k)}p(n))$ algorithm for Vertex Cover-3. Using the results of Section 4, we conclude in Corollary 5.1 that none of these problems has a $O(2^{o(\sqrt{k})}p(n))$ algorithm unless 3-SAT is in DTIME$(2^{o(n)})$.

Finally, in Section 6, we discuss open problems and future directions.

2. Preliminaries

Here we briefly introduce the necessary concepts in the theory of parameterized complexity. For additional information, we refer readers to the comprehensive text on parameterized complexity by Downey and Fellows [16].

To begin, a parameterized problem Π is defined over the set $\Sigma^* \times \mathbb{N}$, where Σ is a finite alphabet and $\mathbb{N}$ in the set of natural numbers. Therefore, each instance of the problem Π is a pair $\langle I, k \rangle$, where k is called the *parameter*. A problem Π is *parameterized tractable* if there is an algorithm running in time $O(f(k)p(|I|))$ that solves the parameterized problem Π for some polynomial p and some recursive function f. The complexity class FPT contains all parameterized tractable problems.

The theory of parameterized complexity defines a variety of reductions that preserve parameterized tractability. Here we employ the standard parameterized m-reduction. Briefly, Π_1 is *parameterized reducible* to Π_2 if there exist functions $g : \mathbb{N} \to \mathbb{N}$ and $f : \Sigma^* \times \mathbb{N} \to \Sigma^*$ such that $\langle x, k \rangle \in \Pi_1 \leftrightarrow \langle f(x,k), g(k) \rangle \in \Pi_2$ and $f(x,k)$ is computable in time $g(k)p(|x|)$ for some polynomial p. Based on this reduction, a hierarchy of increasingly difficult parameterized problems can be defined. Consider the problem of determining whether a circuit of *weft* t has a

satisfying assignment of weight k for some $t \geqslant 1$. Let

$$L_t = \{\langle C_i, k\rangle \mid C_i \text{ is a weft } t \text{ circuit having a weight } k \text{ satisfying assignment}\}$$

and

$$L = \bigcup_{i=1}^{\infty} L_t.$$

Then, $W[t]$ is the class of all problems that are parameterized reducible to L_t, and $W[P]$ is the class of all problems parameterized reducible to L. It is known that $\mathrm{FPT} \subseteq W[1] \subseteq W[2] \subseteq \cdots W[P]$. This is the W-hierarchy. A problem Π is $W[t]$-hard if every problem in $W[t]$ is parameterized reducible to Π and $W[t]$-complete if it is also in $W[t]$.

Many parameterized problems are naturally obtained from optimization problems through parameterizations. Following the earlier work of Cai and Chen [9], we use a standard parameterization of optimization problems. For each optimization problem Π, the *standard parameterized version* of Π is to determine, given an instance I of Π and an integer k, whether the optimal solution cost $\mathrm{OPT}_{\Pi}(I)$ is $\geqslant k$ for maximization problems or $\leqslant k$ in the case of a minimization problems.

An optimization problem Π is said to be parameterized tractable if the standard parameterized version of the problem is parameterized tractable. We also use Π to denote the standard parameterized version of the optimization problem Π wherever it does not cause confusion.

Definition 2.1. Let $s(n)$ be a function. Let Π be a parameterized problem with parameter k. Then $\Pi^{s(n)}$ is the parameterized problem Π with the parameter k replaced by $ks(n)$. $\Pi^{s(n)}$ is called an *extended version* of Π.

In the literature, most parameterized tractability proofs involve explicit constructions of a witness to each positive answer. In particular, the following "stronger" definition of parameterized tractability was introduced in [9].

Definition 2.2. A parameterized problem Π is *parameterized tractable with witness* if there is a $f(k)p(|I|)$-time algorithm that determines the membership of $\langle I, k\rangle$ in Π and also produces a witness to the assertion $\langle I, k\rangle \in \Pi$, for some recursive function f and polynomial p.

In general, we assume the usual definition of parameterized tractability throughout the paper. The stronger definition given above is used in Section 4 to show that L reductions preserve parameterized complexity. This fact is crucial to some of our results in Sections 4 and 5. Note, however, that the close relationship between search and decision means that the terms *parameterized tractable* and *parameterized tractable with witness* are equivalent in many cases. As explained in Sections 4 and 5, this fact allows us to state those results without reference to the witness characterization.

Now we show some properties of the parameterized problems $\Pi^{s(n)}$. First, we note that $s(n)$ can be as large as $o(\log n)$ without significantly changing the parameterized tractability of $\Pi^{s(n)}$. Consider the following technical lemma.

Lemma 2.1. *A parameterized problem is parameterized tractable if it is solvable in time* $O(2^{O(s(n)k)}p(n))$ *for some unbounded and nondecreasing function* $s(n) = o(\log n)$ *and some polynomial p.*

Proof. Define $T(n,k) = O(2^{O(s(n)k)}p(n))$. Let $s(n) = O(\log n/t(n))$ for some unbounded and nondecreasing function $t(n)$. If $k \leqslant t(n)$, then the time $T(n,k) = O(2^{O(\log n)}p(n))$ is a polynomial in n. If $k > t(n)$, then $n \leqslant g(k)$, where $g(k)$ is the inverse function of $t(n)$, and $T(n,k) = O(2^{O(s(g(k))k)}p(n))$, i.e. $T(n,k) = f(k)p(n)$ for some function $f(k)$.

Cai et al. [10] have shown how to efficiently construct such inverse functions using a limited amount of time and space resources. We refer the reader to that paper for further details. □

Theorem 2.1. *Let* Π *be a parameterized problem solvable in time* $O(2^{O(k)}p(n))$ *for some polynomial p. Then for any unbounded nondecreasing function* $s(n) = o(\log n)$, $\Pi^{s(n)}$ *is parameterized tractable.*

Proof. Since $\Pi^{s(n)}$ can be solved in $O(2^{O(s(n)k)}p(n))$ steps via the algorithm for Π, this follows immediately from Lemma 2.1. □

It is natural to ask whether the above theorem holds when $s = \Omega(\log n)$. As we show in Section 3, this is unlikely since it implies that $W[1] = \text{FPT}$. Indeed, $\Pi^{\log n}$ appears to be parameterized intractable for certain problems Π in FPT. Furthermore, the parameterized intractability of $\Pi^{\log n}$ implies a strong lower bound on the running times of parameterized algorithms for Π.

Theorem 2.2. *If parameterized tractable problem* Π *is solvable in time* $O(2^{o(k)}p(n))$ *for some polynomial p, then problem* $\Pi^{\log n}$ *is parameterized tractable.*

Proof. If Π is solvable in time $O(2^{k/t(k)}p(n))$ for some unbounded nondecreasing function t, problem $\Pi^{\log n}$ is solvable in time $O(2^{m/t(m)}p(n))$, where $m = k \log n$ by using the same algorithm. Then

$$m/t(m) = k \log n/t(k \log n) \leqslant k \log n/t(\log n) = s(n)k$$

for some function $s(n) = \log n/t(\log n) = o(\log n)$. Therefore, the running time of this algorithm for $\Pi^{\log n}$ is $O(2^{s(n)k}p(n))$. By Lemma 2.1, the problem $\Pi^{\log n}$ is parameterized tractable. □

3. Subexponential-time parameterized algorithms and W-hierarchy

In this section, we show that there are many parameterized problems in FPT for which the existence of subexponential-time parameterized algorithms is related to the collapse of the W-hierarchy. Our approach will be based on the extended parameterization framework introduced in the previous section.

To begin, we give some definitions for normalized boolean circuits and formulae.

Definition 3.1. Let $t \geqslant 1$ be an integer. A boolean circuit C is called *t-normalized* if

(1) negations occur only at the input literals,
(2) **AND** and **OR** gates are arranged alternatively in levels and gates at level i receive inputs only from level $(i-1)$,
(3) it has $t+1$ levels (t alternations) of gates, including the input literals which are at level 0, and
(4) the output gate (at level t) is an **AND** gate.

A t-normalized circuit is *monotone* if it contains no negations. A t-normalized circuit is *antimonotone* if the 0th level contains only negated literals.

We note that Definition 3.1 is essentially the definition given in [16, p. 284]. Normalized boolean formulae can also be defined in a similar fashion. Let n be the size of a circuit. Let $b(n)$ be function. A normalized circuit has *bottom fan-in* $b(n)$ if the gates at the first level have fan-in bounded by $b(n)$ [11], where n is the total number of inputs to the circuit.

Definition 3.2. Let $t \geqslant 1$. The parameterized problem Half-Weight t-Norm SAT is to determine, given a t-normalized circuit C with k variables, whether there is a *half-weight* satisfying assignment (the weight of the assignment is half of the number of variables) for C, where k is the parameter.

For each $t \geqslant 1$, the problem Half-Weight t-Norm SAT can be solved by a $O(2^k n)$-time parameterized algorithm which tries all possible half-weight assignments to the variables of the circuit, where n the size of the circuit, i.e. the number of gates in the circuit, which includes the input gates.

Note that the extended version Half-Weight t-Norm SAT$^{s(n)}$ of Half-Weight t-Norm SAT is to determine whether there is a *half-weight* satisfying assignment for a given t-normalized circuit with $ks(n)$ variables. Again, n is the size of the circuit. According to Theorem 2.1, for any $s(n) = o(\log n)$, the extended version Half-Weight t-Norm SAT$^{s(n)}$ remains parameterized tractable. However, we will show in the following that for $t \geqslant 2$, Half-Weight t-Norm SAT$^{s(n)}$ is parameterized intractable when $s(n) = \Omega(\log n)$.

Lemma 3.1. *For each* $t \geqslant 1$, Half-Weight $(t+1)$-Norm SAT$^{\log n}$ *is* $W[t]$*-hard.*

Proof. We prove the hardness by constructing reductions to Half-Weight $t+1$-Norm SAT$^{\log n}$ from well-known $W[t]$-hard problems.

According to Downey and Fellows [16, Theorem 12.6], the following problems are complete for $W[t]$ for even values of $t > 1$: Weighted Monotone t-Normalized Satisfiability. This problem is to determine, given a t-normalized monotone boolean formula ϕ and parameter k, whether ϕ has a weight-k satisfying assignment. Similarly, the following problems are complete for $W[t]$ for $t \geqslant 1$ when t is odd: Weighted Antimonotone t-Normalized Satisfiability. In addition, Weighted Antimonotone 2-SAT is complete for $W[1]$. This problem is defined as follows: given a

antimonotone CNF formula ϕ with each clause having at most two literals, and parameter k, determine if ϕ has a weight-k satisfying assignment.

We first consider the case that $t \geqslant 1$ is even. Let ϕ be an instance for the $W[t]$-complete problem Weighted Monotone t-Normalized Satisfiability, i.e., a t-normalized monotone boolean formula. Let $X = (x_1, \ldots, x_{n'})$ be the set of variables in ϕ. We rewrite ϕ as a $(t+1)$-normalized circuit C_ϕ with $2k \log n'$ input variables arranged in k blocks of $2 \log n'$ variables.

Let $Z = (z^{(1)}, \ldots, z^{(k)})$ be the k blocks of input variables for circuit C_ϕ, where for each $i = 1, \ldots, k$,

$$z^{(i)} = (u_1^{(i)}, v_1^{(i)}, \ldots, u_{\log n'}^{(i)}, v_{\log n'}^{(i)})$$

is a vector of $2 \log n'$ boolean variables. In this construction, the block $z^{(i)}$ corresponds to an encoding of the ith true input variable in the formula ϕ. This encoding requires some explanation.

A desired assignment to ϕ will have exactly k variables that are set to true. Let x_j be the ith variable in ϕ that is set to true. This fact will be encoded in the block $z^{(i)}$. Let $B_j = b_1^{(j)} \cdots b_{\log n'}^{(j)}$ be the binary representation of the index j. (Note that the length of B_j is at most $\log n'$ since $1 \leqslant j \leqslant n'$.) To encode that x_j is the ith true variable, the variables in block $z^{(i)}$ will be set such that for $l = 1, \ldots, \log n'$,

(1) $u_l^{(i)} = 1$ and $v_l^{(i)} = 0$ if and only if $b_l^{(j)} = 1$, and
(2) $u_l^{(i)} = 0$ and $v_l^{(i)} = 1$ if and only if $b_l^{(j)} = 0$.

To determine the truth value of x_j in the original formula, it suffices to compute an **OR** of **AND**s using the variables in the k blocks of $2 \log n'$ variables. This circuit is illustrated in Fig. 1.

Now, the reduction from ϕ proceeds as follows. Since t is even and ϕ is monotone, each term T_r at the first level of ϕ is of the form $T_r = \bigvee_{j=1}^{b} x_{r_j}$, where b is the bottom fan-in. Note that T_r is true if and only there is an index r_j corresponding to variable x_{r_j} in ϕ that is set to true. In other words, T_r is true if and only if there is a block $z^{(i)}$, $1 \leqslant i \leqslant k$, representing the index r_j. Therefore, T_r can be

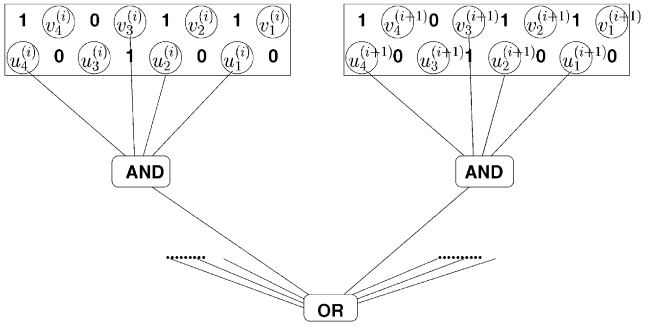

Fig. 1. A circuit to compute the truth value of the variable x_4.

written as

$$T_r = \bigvee_{j=1}^{b}\left(\bigvee_{i=1}^{k}\bigwedge_{l=1}^{\log n'} w_{l,j}^{(i)}\right),$$

where $w_{l,j}^{(i)} = u_l^{(i)}$ if $b_l^{(r_j)} = 1$ in B_{r_j} and $w_{l,j}^{(i)} = v_l^{(i)}$ if $b_l^{(r_j)} = 0$ in B_{r_j}.

To ensure that the blocks $z^{(i)}$ in Z represent mutually distinct indexes, we add the terms

$$M_{i,j} = \bigvee_{l=1}^{\log n'} (u_l^{(i)} \wedge \neg u_l^{(j)}) \vee (\neg u_l^{(i)} \wedge u_l^{(j)}) \vee (v_l^{(i)} \wedge \neg v_l^{(j)}) \vee (\neg v_l^{(i)} \wedge v_l^{(j)})$$

to the boolean formula for all $i,j = 1, \ldots, k$ and $i \neq j$. To ensure, for each $i = 1, \ldots k$, that the values $u_l^{(i)}$ and $v_l^{(i)}$ are exclusive, the term

$$N_i = \bigwedge_{l=1}^{\log n'} (u_l^{(i)} \wedge \neg v_l^{(i)}) \vee (\neg u_l^{(i)} \wedge v_l^{(i)})$$

is also added to the boolean formula.

Now all terms in the formula ϕ are converted into boolean gates in the regular way, resulting in a $(t+1)$-normalized boolean circuit C_ϕ. Note that for every term T_r at the first level of formula ϕ, T_r is true under some weight k assignment $\mathscr{A}(X)$ for variables in X if and only if the corresponding converted gate is true under the assignment for variables in Z that encode $\mathscr{A}(X)$. Therefore, it is not hard to verify that formula ϕ has a weight k satisfying assignment if and only circuit C_ϕ has a weight $k \log n'$ satisfying assignment. Also note that Z contains $2k \log n'$ variables and any valid assignment to C_ϕ is of *half weight*.

Now we consider the case that t is odd. When $t > 1$, we can also convert a t-normalized antimonotone boolean formula to a $(t+1)$-normalized boolean circuit C_ϕ. The construction is similar to the case that t is even. The main part that needs to be treated differently is the representation of each term T_r at the first level of formula ϕ. Since ϕ is antimonotone and t is odd, T_r should be of form $T_r = \bigwedge_{j=1}^{b} \neg x_{r_j}$. Nevertheless, T_r can be written as

$$T_r = \bigwedge_{j=1}^{b}\left(\bigwedge_{i=1}^{k}\bigvee_{l=1}^{\log n'} w_{l,j}^{(i)}\right),$$

where $w_{l,j}^{(i)} = u_l^{(i)}$ if $b_l^{(r_j)} = 0$ in B_{r_j} and $w_{l,j}^{(i)} = v_l^{(i)}$ if $b_l^{(r_j)} = 1$ in B_{r_j}. To complete the construction, each $M_{i,j}$ and N_i need to be converted to an **AND** of **OR**s. This can be done by applying the following generic construction. Given a boolean formula Φ, compute a truth table for $\neg\Phi$. Next, build an **OR** of **AND**s for $\neg\Phi$ directly from the truth table. Build the corresponding **AND** of **OR**s for $\neg(\neg\Phi)$ by applying DeMorgan's law. Notice that the resulting circuits for $M_{i,j}$ and N_i may have $O(n'^4)$ and $O(n'^2)$ gates, respectively, since $M_{i,j}$ has $4 \log n'$ inputs and N_i has $2 \log n'$ inputs.

When $t = 1$, we can also convert an antimonotone 2-CNF formula ϕ into a 2-normalized boolean circuit C_ϕ in roughly the same way. For each clause of the form $\Phi = (\neg x_a \vee \neg x_b)$ in ϕ, we

convert it to a boolean formula of the form

$$\Phi' = \left(\left(\bigwedge_{i=1}^{k} \bigvee_{l=1}^{\log n'} w_{l,a}^{(i)} \right) \vee \left(\bigwedge_{i=1}^{k} \bigvee_{l=1}^{\log n'} w_{l,b}^{(i)} \right) \right).$$

Then, we build an **OR** of **AND**s for $\neg\Phi'$ via its truth table. Applying DeMorgan's law gives an **AND** of **OR**s for Φ' with $O(n'^4)$ clauses. Applying this construction to each clause gives a circuit that is an AND of ORs, which is essentially a new boolean formula in CNF.

To complete the proof, we need to make an adjustment to the number of variables. Our construction creates a circuit with $2k\lceil \log n' \rceil$ variables. However, this is not technically an instance of Half-Weight $(t+1)$-Norm SAT$^{\log n}$ because n' is the number of variables in the original circuit, not the number of gates in the circuit. We give the construction for Half-Weight 2-Norm SAT. Constructions for the other problems are similar.

Let M be the number of gates in C_ϕ. To fix C_ϕ to match the definition, we add $2Q$ new variables y_i and $3Q$ gates to C_ϕ. For each i ranging from 1 to Q, we add the circuit $\neg y_{2i-1} \vee \neg y_{2i}$, and we connect the result of this circuit to the AND gate at the top level. This adds $2Q$ new variables and $3Q$ new gates to the circuit. Notice that a satisfying assignment to the resulting circuit can have at most one half of the variables y_i set to true. So, the resulting circuit C'_ϕ has a half-weight satisfying assignment if and only if C_ϕ has a half-weight satisfying assignment. So, it suffices to find a value for Q such that $2k\lceil \log M + 3Q \rceil = 2k\lceil \log n' \rceil + 2Q$. This occurs when $Q = k(\log\lceil M + 3Q \rceil - \lceil \log n' \rceil)$. For increasing integer values of Q, the right-hand side of this equation either remains the same, or increases by k. Since the right-hand side grows more slowly than Q, at some point these two values must be equal. Since $M \geqslant n' \geqslant k \geqslant \log k$, this occurs before $Q = 4kM$ if M is at least four, irrespective of the value of n'. Finding the exact value of Q can be done by standard equation solving techniques (e.g., bisection) in polynomial-time. Now, the original C_ϕ was satisfiable only when half of the variables are set to true, so ϕ has a weight k satisfying assignment if and only if C'_ϕ has an satisfying assignment of where exactly one half of the variables are set to true. □

Theorem 3.1. *For each* $t \geqslant 1$, *unless* $W[t] = \text{FPT}$, *parameterized problem* Half-Weight $(t+1)$-Norm SAT *cannot be solved by parameterized algorithms of running time* $O(2^{o(k)}p(n))$ *for any polynomial* p.

Proof. If Half-Weight t-Norm Sat is solvable in time $O(2^{o(k)}p(n))$ for some polynomial p, by Theorem 2.2, the extended parameterized version Half-Weight $(t+1)$-Norm SAT$^{\log n}$ is parameterized tractable. Together with Lemma 3.1, this leads to $W[t] = \text{FPT}$. □

Based on the proof of Lemma 3.1, the circuit C_ϕ constructed is actually of $O(\log n)$ bottom fan-ins and every pair of variables $u_l^{(i)}, v_l^{(i)}$ take exclusive values. Define Half Weight SAT-m to be the following variant of problem SAT: given a CNF boolean formula with $2k$ input variables $\{x_1, x_2, \ldots, x_{2k-1}, x_{2k}\}$ and $O(m)$ bottom fan-in (where k is the parameter), determine whether C has a satisfying assignment in which exactly one of (x_{2i-1}, x_{2i}) is true for all $i = 1, \ldots, k$.

Corollary 3.1. *Unless* $W[1] = \text{FPT}$, *the parameterized problem* Half Weight SAT-log n *cannot be solved in time* $O(2^{o(k)}p(n))$ *for any polynomial* p.

This corollary complements the earlier result by Abrahamson et al. [1] that $W[1] = \text{FPT}$ implies $\text{3-SAT} \in \text{DTIME}(2^{o(n)})$. It is not known, however, whether $\text{3-SAT} \in \text{DTIME}(2^{o(n)})$ would imply $W[1] = \text{FPT}$. In particular, in the next section, we will show that the existence of subexponential-time parameterized algorithms for many important optimization problems in the class Max-SNP also hinges on whether $\text{3-SAT} \in \text{DTIME}(2^{o(n)})$.

4. Subexponential-time algorithms for MAX-SNP: the hardness

As noted in [16], a number of NP-complete problems such as Vertex Cover, Vertex Cover-B, Max Sat, Max c-Sat, Max k-Cut, and Independent Set for bounded degree graphs are parameterized tractable. In particular, each of these problems can be solved in time $O(2^{O(k)}p(n))$ for some polynomial p. It is natural to ask whether the running times of parameterized algorithms for these problems can be significantly improved. In this section, we work towards answering these questions through an investigation of parameterized versions of Max-SNP-hard optimization problems.

In this section, we employ the work of Impagliazzo et al. [20] to connect the parameterized complexity of the Max-SNP-hard problems to the classical complexity of 3-SAT. We begin by summarizing the necessary details from their work. Briefly, a *search problem* consists of finding, when given an input x, a witness y of size $\leqslant m(x)$ that satisfies a polynomial-time computable relation $R(x, y)$. In this context, $m(x)$ is the *complexity parameter*. A search problem is said to be *subexponential-time computable* if it can be solved in $p(|x|)2^{\varepsilon m(x)}$ for some fixed polynomial p and every $\varepsilon > 0$.

The work of Impagliazzo et al. [20] defines a reduction family, called the **SERF** *reducibility*, that preserves subexponential-time computation among search problems and their associated complexity parameters. As noted in the paper, some NP-completeness reductions preserve subexponential time computability among search problems as well. In fact, any many-one reduction f from S_1, m_1 to S_2, m_2 that satisfies $m_2(f(x)) = O(m_1(x))$ preserves subexponential time computability. Such reductions are referred to a strong many-one reductions. Strong many-one reductions are a special case of the SERF reducibility.

Impagliazzo et al. [20] define a notion of completeness for the class SNP under the SERF reducibility, and they prove that both 3-SAT with the complexity parameter n (the number of variables) and 3-SAT with the complexity parameter m (the number of clauses) are complete for SNP under the SERF reducibility. It follows that 3-SAT has a subexponential time algorithm with the complexity parameter n if and only if 3-SAT has a subexponential-time algorithm with the complexity parameter m. This relationship is true for all search problems that are complete for SNP under the SERF reducibility.

The class Max-SNP was introduced by Papadimitriou and Yannakakis [24] to capture a collection of optimization problems. For the purpose of investigating the approximability of these problems, the following approximation-preserving reduction was introduced.

Definition 4.1 (Papadimitriou and Yannakakis [24]). Let Π_1 and Π_2 be two optimization problems with cost functions f_1 and f_2. Π_1 *L*-reduces to Π_2 if there are two polynomial time algorithms A and B and two constants $\alpha, \beta > 0$ such that for each instance I_1 of Π_1,

(1) the algorithm A produces an instance $I_2 = A(I_1)$ such that $\mathrm{OPT}_{\Pi_2}(I_2) \leqslant \alpha \mathrm{OPT}_{\Pi_1}(I_1)$, and
(2) given any solution S_2 for I_2 with cost $f_2(I_2, S_2)$, algorithm B produces a solution S_1 for I_1 with cost $f_1(I_1, S_1)$ such that $|\mathrm{OPT}_{\Pi_1}(I_1) - f_1(I_1, S_1)| \leqslant \beta |\mathrm{OPT}_{\Pi_2}(I_2) - f_2(I_2, S_2)|$.

It is known from the work of Cai and Chen [9] that the standard parameterized versions of all maximization problems in the Max-SNP are parameterized tractable. The proof of this earlier result shows that *L* reductions preserve parameterized tractability. Here we provide a more detailed account of how *L* reductions preserve parameterized tractability among the standard parameterized versions of optimization problems. In particular, we show that *L*-reductions preserve subexponential-time computability.

Lemma 4.1. *Let Π_1 and Π_2 be two optimization problems such that Π_1 L-reduces to Π_2, and assume that the cost function for Π_2 is integer-valued. If Π_2 is solvable with witness in time $O(f(k)p(n))$ for some recursive function f and polynomial p then Π_1 can be solved in time $O(kf(O(k))q(n))$ for some q polynomial.*

Proof. We prove the lemma for the case when Π_1 and Π_2 are maximization problems. The proofs for the other three cases are similar.

Assume that Π_1 *L* reduces to Π_2 via the algorithms A and B running in time $t(n)$ and $q(n)$, respectively. Let α and β be the constants associated with this *L* reduction. Now, assume that Π_2 is computable in $O(f(k)p(n))$ steps for some recursive function f and polynomial p. We can decide Π_1 as follows. Given an instance $\langle I_1, k \rangle$ of Π_1, compute $I_2 = A(I_1)$ and run the algorithm for Π_2 on $\langle I_2, \alpha(k+1) \rangle$. If the answer is yes, then $\alpha \mathrm{OPT}_{\Pi_1}(I_1) \geqslant \mathrm{OPT}_{\Pi_2}(I_2) \geqslant \alpha(k+1)$ and hence $\mathrm{OPT}_{\Pi_1}(I_1) \geqslant k$. If the answer is no, then we run the algorithm for Π_2 on the pairs $\langle I_2, \alpha(k+1)-1 \rangle, \ldots, \langle I_2, 0 \rangle$ until the answer is yes. Because $\mathrm{OPT}_{\Pi_2}(I_2)$ is integer-valued, the answer to at least one of these will be yes. On the first such yes value, the algorithm for Π_2 must produce a witness solution s_2 satisfying $f_2(I_2, s_2) = \mathrm{OPT}_{\Pi_2}(I_2)$. By applying the function B to s_2, we get a solution to I_1 satisfying $|\mathrm{OPT}_{\Pi_1}(I_1) - f_1(I_1, s_1)| \leqslant \beta |\mathrm{OPT}_{\Pi_2}(I_2) - f_2(I_2, s_2)| = 0$, and hence s_1 is the optimal solution to I_1. In this case, we can use the value $f_1(I_1, s_1)$ to determine whether $\mathrm{OPT}_{\Pi_1}(I_1) \geqslant k$.

Notice that the algorithm for Π_2 is called at most $\alpha(k+1)$ times and the algorithms A and B are called only once. Hence, the time bound stated in the lemma for the algorithm can be easily verified. □

Because Max 3-SAT is Max-SNP-complete [24] and parameterized tractable with witness [9], we obtain the following result through Lemma 4.1.

Corollary 4.1. *Each optimization problem in the class* Max-SNP *is solvable in time $O(2^{O(k)}p(n))$ for some polynomial p.*

Lemma 4.1 also implies that *L*-reductions preserve subexponential parameterized complexity.

Theorem 4.1. *Let Π_1 be a* Max-SNP*-hard* (*under L-reductions*) *optimization problem with an integer-valued cost function. If Π_1 is solvable with witness in time $O(2^{o(k)}p(n))$ for some polynomial p, then each optimization problem Π_2 in* Max-SNP *is solvable in time $O(2^{o(k)}q(n))$ for some polynomial q.*

Proof. Since Π_1 is solvable with witness in time $O(2^{o(k)}p(n))$ for some polynomial p and each problem Π_2 in Max-SNP is L-reducible to Π_1, it follows from Lemma 4.1 that Π_2 can be solved in $O(2^{o(k)}q(n))$ steps for some polynomial q. □

To show the hardness to obtain $O(2^{o(k)}p(n))$-time parameterized algorithms (for any polynomial p), for Max-SNP-hard problems, we first establish the close relationship between the parameterized complexity of Vertex Cover-3, the vertex cover problem on graphs with degrees bounded by 3, and the classical complexity of 3-SAT. Consider the following theorem.

Theorem 4.2. *If parameterized problem* Vertex Cover-3 *is solvable in time $O(2^{o(k)}p(n))$ for some polynomial p, then* 3-SAT *is solvable in time $O(2^{o(n)})$, where n is the number of variables in each* 3-CNF *formula.*

Proof. Notice that if the parameterized version of Vertex Cover-3 is solvable in $O(2^{o(k)}p(n))$ for some polynomial p, then Vertex Cover-3 can easily be solved in time $O(p(n)2^{o(n)})$ steps since $k \leqslant n$. To complete the proof, it suffices to show that such an algorithm for Vertex Cover-3 implies a fast algorithm for 3-SAT.

While not explicitly proven in [20], it is the case that Vertex Cover-3 is SNP-complete under SERF reductions with the parameter n, the number of vertices. To see this, notice that there exists a strong many-one reduction from 3-SAT with the parameter m, the number of clauses, to Vertex Cover-3 with the parameter n [19, Theorem 2.6]. Since a strong many-one reduction is a special case of the SERF reducibility, it follows that Vertex Cover-3 is SNP-complete under SERF reductions.

Since 3-SAT with the parameter n, the number of variables is also SNP-complete under SERF reductions, Theorem 1 in Section 3 of [20] allows us to conclude that 3-SAT can be computed in time $p(|x|)2^{\varepsilon n}$ for every $\varepsilon > 0$. While the definition of subexponential-time computation in [20] includes that possibility that a different algorithm is needed for each $\varepsilon > 0$, the sparsification lemma of Impagliazzo et al. [20, Theorem 1, Corollary 1 of Section 2] can be used to strengthen the result to able to conclude that 3-SAT $\in$ DTIME$(2^{o(n)})$. □

Next, we prove the reverse direction. Consider the following lemma.

Lemma 4.2. *There is a polynomial-time reduction that given a* 3*-degree graph $G = (V, E)$ and a parameter k, produces a* 3-CNF *formula ϕ with at most $c|V|$ variables such that G has a vertex cover of size k if and only if ϕ is satisfiable.*

Proof. Whether the graph G has a vertex cover of size k can be reformulated as the question of whether boolean circuit C has a weight k satisfying assignment, where $C = \bigwedge_{\langle v_i, v_j \rangle \in E} (x_i \vee x_j)$,

consisting of $|V|$ input variables $x_1, \ldots, x_n$ and $|E|+1$ gates such that $x_i = 1$ if and only if $v_i \in V$ is in the vertex cover.

To ensure that only weight-k assignments are valid assignments, we use a subcircuit D to verify if there are exactly k 1's in the input vector $[x_1, \ldots, x_n]$. The subcircuit D consists of $(n-1)$ binary adders to count the number of 1's in the input vector and a component A to compare the outcome of the count with value k in binary. These adders are structured as a binary tree as shown in Fig. 2. Each adder at level h takes two binary numbers of length h produced by some two adders at level $h-1$ and produces the sum in binary of length $h+1$ to the next level.

It is clear that each adder can be implemented using boolean gates and the number of boolean gates needed in each adder is $O(h)$, which is linear in the length h of the two inputs to the adder. Therefore, the total number of boolean gates needs for these adders is at most

$$\sum_{h=1}^{\lceil \log n \rceil} \left\lfloor \frac{n}{2^h} \right\rfloor dh \leqslant dn \sum_{h=1}^{\lceil \log n \rceil} h\left(\frac{1}{2}\right)^h \leqslant dn \sum_{h=0}^{\infty} h\left(\frac{1}{2}\right)^h = dn \frac{1/2}{(1-1/2)^2} = 2dn = O(n)$$

for some constant d. Moreover, the number of boolean gates needed to implement component A is $O(\log n)$ because $k \leqslant n$.

Let C' be the circuit C embedded with subcircuit D so that C has a weight-k satisfying assignment if and only if C' is satisfiable. Note that the total number of gates in circuit C' is bounded by $O(n) + |E| = O(n)$ since G is a degree-3 graph. As given in the textbook by Cormen et al. [14], there is a polynomial-time reduction that reduces a boolean circuit to a 3-CNF formula in which the number variables increased is linear in the total number of gates in the circuit. Using this technique, we can reduce the circuit C' to a 3-CNF formula ϕ with $O(n)$ variables. □

To conclude that Vertex Cover-3 has a $O(2^{o(k)}p(n))$ parameterized algorithm if and only 3-SAT has a $O(2^{o(n)})$ algorithm, we employ a reduction to kernel. Here we use the following observation by Chen et al. [13, Theorem 2.2]. Their result combines the earlier work of Nemhauser and Trotter

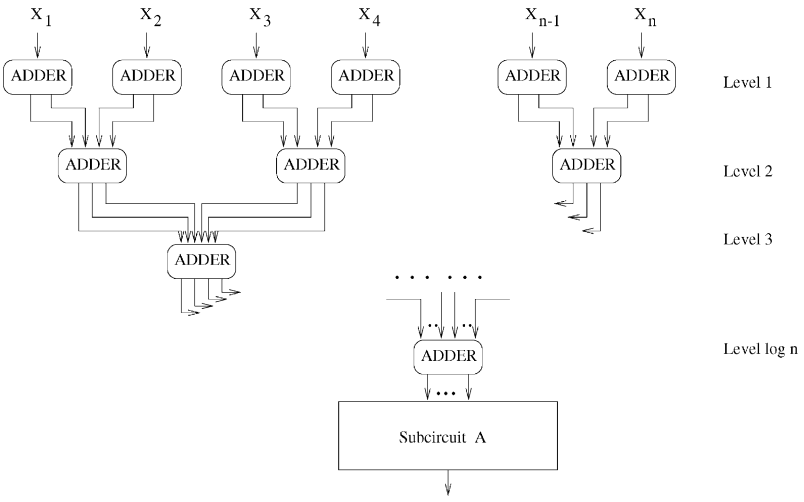

Fig. 2. Circuitry to verify a weight k assignment.

[21] and Buss [8] to achieve a strong reduction to kernal for Vertex Cover. This strong reduction to kernel will also be used in the next section.

Proposition 4.1 (Chen et al. [13]). *There exists an algorithm running in time $O(kn + k^3)$ that when given $\langle G, k \rangle$ produces $\langle G_1, k_1 \rangle$ such that G has a vertex cover of size k if and only if G_1 has a vertex cover of size k_1, where G_1 has $\leqslant 2k_1$ vertices, $k_1 \leqslant k$, and G_1 is a subgraph of G.*

Combining Theorem 4.2, Lemma 4.2, and Proposition 4.1, we have the following result.

Theorem 4.3. Vertex Cover-3 *can be solved in time $O(2^{o(k)}p(n))$ for some polynomial p if and only if* 3-SAT $\in$ DTIME$(2^{o(n)})$, *where n is the number of variables.*

Proof. The first direction comes directly from Theorem 4.2. For the second direction, assume that 3-SAT $\in$ DTIME$(2^{o(n)})$. Then, by Lemma 4.2 there exists a $O(2^{o(|V|)})$ time algorithm for Vertex Cover-3. Combining this algorithm with the strong reduction to kernal given by Proposition 4.1, gives a $O(2^{o(k)}p(n))$ algorithm for Vertex Cover-3. □

By Theorems 4.1 and 4.3 and the fact that Vertex Cover-3 can be solved in time $O(2^{o(k)}p(n))$ if and only if it can be solved in time $O(2^{o(k)}q(n))$ with witness, we obtain the following connecting result.

Theorem 4.4. *Each* Max-SNP-*complete problem Π can be solved in time $O(2^{o(k)}p(n))$ with witness for some polynomial p if and only if* 3-SAT $\in$ DTIME$(2^{o(n)})$, *where n is the number of variables.*

Since the proof of above theorem relies on Theorem 4.1, it does not appear that we can easily remove the word "witness" from the statement of our main result. However, in practice, it is often the case that the complexities of decision problems and their witness versions are closely related [6]. Hence, Theorem 4.4 gives the following immediate corollaries.

Corollary 4.2. *The parameterized problems* Max c-SAT, Vertex Cover-B, Independent Set-B, *and* Dominating Set-B *can be solved in time $O(2^{o(k)}p(n))$ for some polynomial p if and only if* 3-SAT $\in$ DTIME$(2^{o(n)})$, *where n is the number of variables.*

Corollary 4.3. *The parameterized problems* Max SAT, Vertex Cover, and Max c-Cut *cannot be solved in time $O(2^{o(k)}p(n))$ for any polynomial p unless* 3-SAT $\in$ DTIME$(2^{o(n)})$, *where n is the number of variables.*

Therefore, the existence of subexponential-time parameterized algorithms for any Max-SNP-hard optimization problem relates to the long-standing open question concerning the membership of 3-SAT in DTIME$(2^{o(n)})$. Note that while it is widely believed that 3-SAT $\notin$ DTIME$(2^{o(n)})$, Abrahamson et al. [1] have shown that $W[1] = \text{FPT}$ would also imply 3-SAT $\in$ DTIME$(2^{o(n)})$.

5. Subexponential-time lower bounds for planar problems

In contrast to hardness results shown in Sections 3 and 4, a number of parameterized problems do admit subexponential-time parameterized algorithms. For example, the recent and somewhat surprising work by Alber et al. [2,3] has shown that many problems on planar graphs, such as Planar Dominating Set and Planar Vertex Cover, have parameterized algorithms of running time $O(2^{O(\sqrt{k})}p(n))$. Since each of these problems is NP-complete, it is natural to ask if the running time for these new parameterized algorithms can be substantially improved. Results and techniques from previous sections will be applied to answer this question. We will show that the consequences of these problems having $O(2^{o(\sqrt{k})}p(n))$-time parameterized algorithms also relate directly to the classical and parameterized complexity of a variety of problems.

We begin by relating the parameterized complexity of certain problems on planar graphs to the parameterized complexity of Vertex Cover-3.

Lemma 5.1. *If* Planar Vertex Cover, Planar Independent Set, Planar Dominating Set *or* Planar Red/Blue Dominating Set *has a parameterized algorithm running in time* $O(2^{o(\sqrt{k})}p(n))$ *for some polynomial* p, *then* Vertex Cover-3 *has a* $O(2^{o(k)}p(n))$*-time parameterized algorithm.*

Proof. We first prove the result for Planar Vertex Cover. Let A be a $O(2^{o(\sqrt{k})}p(n))$-time algorithm for Planar Vertex Cover. We construct an algorithm B for Vertex Cover on graphs bounded by degree 3 that operates as follows. Let $\langle G, k\rangle$ be an instance of Vertex Cover-3. Using the algorithm given in Proposition 4.1, we transform $\langle G, k\rangle$ into $\langle G_1, k_1\rangle$. Notice that since G_1 is a subgraph of G, each vertex in G_1 has degree bounded by 3. Next, we transform $G_1 = (V_1, E_1)$ into a planar graph $G_2 = (V_2, E_2)$ as follows. Draw the graph G_1 on the plane. At each crossover point defined by a pair of edges (u_1, v_1) and (u_2, v_2), insert the widget given in Fig. 3. This construction produces a planar graph G_2. Furthermore, as shown by Garey et al. [19], G_2 has a vertex cover of size $k_2 = k_1 + 13d$ if and only if G_1 has a vertex cover of size k_1, where d is the number of crossover points. Since $|V_1| \leqslant 2k$ and each vertex has degree bounded by 3, there are at most $3k$ edges in G_1 and hence at most $(3k)^2 = O(k^2)$ crossover points. It follows that $k_2 = O(k^2)$. To complete the algorithm B, simply apply the algorithm A to the instance $\langle G_2, k_2\rangle$ and return the result. Since $k_2 = O(k^2)$, the running time of the algorithm B is $O(2^{o(k)} \cdot p(n))$.

To see that same result holds for Planar Independent Set, note that $G = (V, E)$ has a vertex cover of size k if and only G has an independent set of size $|V| - k$. Since G_2 has

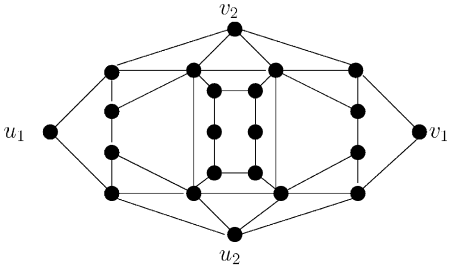

Fig. 3. A widget to remove crossovers.

$|V_1| + 22d = O(k^2)$ vertices, we can solve the original of instance of Vertex Cover-3 by solving Planar Independent Set on the instance $\langle G_2, k_2' \rangle$, where $k_2' = |V_1| + 22d - k_2 = O(k^2)$. The running time of this new algorithm for Vertex Cover-3 is $O(2^{o(k)} \cdot p(n))$.

For Planar Red/Blue Dominating Set, convert the planar graph G_2 to a planar graph G_3 by adding a new vertex v_e for each edge $e = (u, v)$ in G_2 and adding edges from both u and v to v_e. The graph G_3 is also planar. Place the original vertices from V_2 into the "red" set and place the new vertices $v_{(u,v)}$ into the "blue" set. Then, G_2 has a vertex cover of size k_2 if and only if there exists a set of k_2 vertices from the "red" set in G_3 that dominates all the vertices in the "blue" set. So, if we apply a $O(2^{o(\sqrt{k})}p(n))$ for Planar Red/Blue Dominating Set to $\langle G_3, k_2 \rangle$ as our final step, we get a $O(2^{o(k)} \cdot p(n))$ time algorithm for Vertex Cover-3.

For Planar Dominating Set, convert the planar graph G_2 to a planar graph G_4 by adding $k_2 + 1$ new vertices $e_1, \ldots, e_{k_2+1}$ for each edge $e = (u, v)$ in G_2 and adding edges from both u and v to each e_i. Without loss of generality, assume that there are no isolated vertices in G_2. Now, since G_2 is planar, it is easy to see that G_4 is also planar. Moreover, it is easy to show that G_2 has a vertex cover of size k_2 if and only if G_4 has a dominating set of size k_2. To see this, notice that a vertex cover of size k_2 of G_2 is a dominating set of G_4 since if u covers the edge $e = (u, v)$, then u dominates v and all the vertices $e_1, \ldots, e_{k_2+1}$. Since every edge is covered by these k_2 vertices and G_2 contains no isolated vertices, these k_2 vertices must be a dominating set in G_4. Similarly, assume that G_4 has a dominating set of size k_2. Then, for each edge $e = (u, v)$ in G_2 either u or v must be in such a dominating set. If not, then all vertices $e_1, \ldots, e_{k_2+1}$ must be in the dominating set. This is a contradiction. Hence, such a dominating set in G_4 is a vertex cover in G_2. So, if we apply a $O(2^{o(\sqrt{k})}p(n))$ for Planar Dominating Set to $\langle G_4, k_2 \rangle$ as our final step, we get a $O(2^{o(k)} \cdot p(n))$ time algorithm for Vertex Cover-3. □

By combining our earlier results relating the complexity of Vertex Cover-3 to 3-SAT with Lemma 5.1, we obtain the following conditional lower bounds.

Theorem 5.1. Planar Vertex Cover, Planar Independent Set, Planar Dominating Set, *and* Planar Red/Blue Dominating Set *do not have* $O(2^{o(\sqrt{k})}p(n))$*-time parameterized algorithms for any polynomial p unless* Max SNP*-complete problems* Max c-SAT, Vertex Cover-*B*, Independent Set-*B*, *and* Dominating Set-*B have* $O(2^{o(k)}q(n))$*-time parameterized algorithms for some polynomial q*.

Corollary 5.1. *Unless* 3-SAT $\in$ DTIME$(2^{o(n)})$, Planar Vertex Cover, Planar Independent Set, Planar Dominating Set, *and* Planar Red/Blue Dominating Set *do not have* $O(2^{o(\sqrt{k})}p(n))$*-time parameterized algorithms for any polynomial p*.

6. Conclusion and open problems

We have shown that the existence of subexponential-time parameterized algorithms for some problems within FPT implies the collapse of the W-hierarchy at various levels. In addition, we

have provided some evidence that $O(2^{o(k)}p(n))$ algorithms for Vertex Cover and other Max-SNP-hard problems do not exist and that $O(2^{o(\sqrt{k})}p(n))$ algorithms do not exist for some problems on planar graphs such as Planar Vertex Cover by relating the existence of such algorithms to whether 3-SAT can be solved in DTIME$(2^{o(n)})$. The central open question in our work is whether the existence of subexponential-time parameterized algorithms for problems such as Vertex Cover would also imply a collapse in the W-hierarchy. While this result was claimed in [12], a technical error in the conference paper [12, Lemma 3, Theorem 4] leaves this question open.

Our work provides a outline of an approach to solve this open problem: simply prove that Vertex Cover$^{\log n}$ or some other $\log n$ version of a Max-SNP-hard problem is $W[1]$-hard. By Theorem 2.2, the existence of a subexponential-time parameterized algorithm for Vertex Cover would imply that Vertex Cover$^{\log n}$ is parameterized tractable and hence that $W[1] = \text{FPT}$. So, it appears that an investigation of the parameterized complexity of Vertex Cover$^{\log n}$ may be worthwhile. In a recent development, Fellows [18] has shown that Vertex Cover$^{\log n}$ is contained in $W[1]$.

Acknowledgments

The authors thank Jochen Alber, Jianer Chen, Mike Fellows, Rod Downey, Frances Rosamond, Ken Regan, Martin Grohe, Gerd Woeginger, and others at the workshop on Parameterized Complexity at Schloss Dagstuhl for their valuable comments on a preliminary version of this work [12].

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. Bodlaender, H. Fernau, R. Niedermeier, Fixed parameter algorithms for planar dominating set and related problems, in: Proceedings of the Seventh Scandinavian Workshop on Algorithm Theory (SWAT 2000), Lecture Notes in Computer Science, Vol. 1851, Springer, Berlin, 2000, pp. 97–110.
[3] J. Alber, H. Fernau, R. Niedermeier, Parameterized complexity: exponential speed-up for planar graph problems, in: Proceedings of the Twenty-Eighth International Colloquium on Automata, Languages and Programming (ICALP 2001), Lecture Notes in Computer Science, Vol. 2076, Springer, Berlin, 2001, pp. 261–272.
[4] J. Alber, J. Gramm, R. Niedermeier, Faster exact algorithms for hard problems: a parameterized point of view, Discrete Math. 229 (2001) 3–27.
[5] R. Balasubramanian, M.R. Fellows, V. Raman, An improved fixed parameter algorithm for vertex cover, Inform. Process. Lett. 65 (1998) 163–168.
[6] M. Bellare, S. Goldwasser, The complexity of decision versus search, SIAM J. Comput. 23 (1) (1994) 97–119.
[7] J.F. Buss, J. Goldsmith, Nondeterminism in P, SIAM J. Comput. 22 (1993) 560–572.
[8] S. Buss, 1989. Algorithm attributed to S. Buss by J. Buss and J. Goldsmith [7, p. 563].
[9] L. Cai, J. Chen, On fixed-parameter tractability and approximability of NP optimization problems, J. Comput. System Sci. 54 (3) (1997) 465–474.
[10] L. Cai, J. Chen, R. Downey, M. Fellows, On the structure of parameterized problems in NP, Inform. Comput. 123 (1995) 38–49.
[11] L. Cai, J. Chen, J. Håstad, Circuit bottom fan-in and computation power, SIAM J. Comput. 27 (1998) 341–355.

[12] L. Cai, D. Juedes, Subexponential-time parameterized algorithms collapse the W-hierarchy, in: Proceedings of the 28th International Colloquium on Automata, Languages, and Programming, Springer, Berlin, 2001, pp. 273–284.
[13] J. Chen, I. Kanj, W. Jia, Vertex cover: further observations and further improvements, J. Algorithms 41 (2001) 280–301.
[14] T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein, Introduction to Algorithms, 2nd Edition, McGraw Hill, New York, 2001.
[15] R.G. Downey, M.R. Fellows, Parameterized computational feasibility, in: Proceedings of Feasible Mathematics, Vol. II, Birkhauser, Basel, 1995, pp. 219–244.
[16] R.G. Downey, M.R. Fellows, Parameterized Complexity, Springer, New York, 1999.
[17] R.G. Downey, M.R. Fellows, U. Stege, Parameterized complexity: a framework for systematically confronting computational intractability, in: Contemporary Trends in Discrete Mathematics: From DIMACS to DIMATIA to the Future, AMS-DIMACS Proceeding Series, Vol. 49, American Mathematical Society, Providence, RI, 1999, pp. 49–99.
[18] M. Fellows, Personal Communication, 2001.
[19] M.R. Garey, D.S. Johnson, L. Stockmeyer, Some simplified NP-complete graph problems, Theoret. Comput. Sci. 1 (1976) 237–267.
[20] R. Impagliazzo, R. Paturi, F. Zane, Which problems have strongly exponential complexity? in: Proceedings of the 39th Symposium on Foundations of Computer Science, IEEE Computer Society Press, Silver Spring, MD, 1998, pp. 653–664.
[21] G.L. Nemhauser, L.E. Trotter, Vertex packings: structural properties and algorithms, Math. Program. 8 (1975) 232–248.
[22] R. Niedermeier, P. Rossmanith, Upper bounds for vertex cover further improved, in: Proceedings of the 16th Annual Symposium on Theoretical Aspects of Computer Science, Lecture Notes in Computer Science, Vol. 1563, Springer, Berlin, 1999, pp. 561–570.
[23] R. Niedermeier, P. Rossmanith, A general method to speed up fixed-parameter-tractable algorithms, Inform. Process. Lett. 73 (2000) 125–129.
[24] C.H. Papadimitriou, M. Yannakakis, Optimization, approximation, and complexity classes, J. Comput. Systems Sci. 43 (1991) 425–440.