A note on parameterized exponential time complexity

Jianer Chen* Iyad A. Kanj† Ge Xia*

Abstract

In this paper we define the notion of an $f(k)$-uniform parameterized exponential time scheme. We show that a problem can be solved in parameterized $O(2^{o(f(k))}p(n))$ time if and only if it has an $f(k)$-uniform parameterized exponential time scheme (p is a polynomial). We then illustrate how our formulation can be used to show that special instances of parameterized NP-hard problems are as difficult as the general instances. For example, we show that the Planar Dominating Set problem on degree-3 graphs can be solved in $O(2^{o(\sqrt{k})}p(n))$ parameterized time if and only if the general Planar Dominating Set problem can. Apart from their complexity theoretic implications, our results have some interesting algorithmic implications as well.

Key words. parameterized complexity, subexponential time complexity, parameterized algorithms, exact algorithms

1 Introduction

Parameterized complexity theory [7] was motivated by the observation that many important NP-hard problems in practice are associated with a parameter whose value usually falls within a small or a moderate range. Thus, taking the advantage of the small size of the parameter may significantly speedup the computation. Formally, a *parameterized problem* consists of instances of the form (x, k), where x is the *problem description* and k is an integer called the *parameter*. A parameterized problem is *fixed parameter tractable* if it can be solved by an algorithm of running time $O(f(k)n^c)$, where f is a function independent of $n = |x|$ and c is a constant.

A parameterized problem is solvable in *parameterized subexponential time* if it can be solved in time $O(2^{f(k)}p(n))$, where $f(k) \in o(k)$ is a function, and $p(n)$ is a polynomial in the input size n. Very few parameterized NP-hard problems are known to be solvable in parameterized subexponential time. Alber et al. [2] gave parameterized subexponential time algorithms for the Planar Vertex Cover, Planar Independent Set, and Planar Dominating Set problems that run in $O(2^{c\sqrt{k}}n)$, where c is a constant. In particular, improving the upper bounds on the running time of subexponential time algorithms for Planar Dominating Set has been receiving a lot of attention [2, 8, 13]. Currently, the most efficient algorithm for Planar Dominating Set is by Fomin and Thilikos, and runs in $O(2^{15.13\sqrt{k}}n)$ time [8].

On the other hand, deriving lower bounds on the precise complexity of parameterized NP-hard problems has also started attracting more and more attention [4, 12]. Cai and Juedes [4] showed that Planar Vertex Cover, Planar Independent Set, and Planar Dominating Set cannot

*Supported in part by NSF under the grant CCR-0000206. Department of Computer Science, Texas A&M University, College Station, TX 77843-3112. Email: {gexia,chen}@cs.tamu.edu.

†School of CTI, DePaul University, 243 S. Wabash Avenue, Chicago, IL 60604-2301. Email: ikanj@cs.depaul.edu.

be solved in $O(2^{o(\sqrt{k})}p(n))$ time unless 3-SAT is in DTIME($2^{o(n)}$) time, a consequence that seems unlikely according to the common belief among researchers in the field. This basically shows that Alber's et al.'s results [2] are tight up to a constant factor in the exponent. This line of research parallels the one in classical exponential time complexity by Impagliazzo, Paturi, and Zane [10] in which they introduced the concept of SERF-reduction to show that many well-known NP-hard problems are SERF-complete for the class SNP [10]. This implies that if any of these problems is solvable in subexponential time, then so are all problems in the class SNP. A consequence that seems quite unlikely.

In this paper we define the notion of an *$f(k)$-uniform parameterized exponential time scheme*. Let $f(k)$ be an unbounded nondecreasing function of k. A parameterized problem Q is said to have an *$f(k)$-uniform parameterized exponential time scheme* if there exists a parameterized algorithm A for Q, such that on any given instance (x, k) of Q and any given constant $\delta > 0$, the running time of the algorithm A is bounded by $O(2^{\delta f(k)}p(|x|))$, where p is a polynomial. We show that a parameterized problem Q admits an $f(k)$-uniform parameterized exponential time scheme if and only if Q can be solved in $O(2^{o(f(k))}q(n))$ parameterized time, where q is a polynomial. This notion of equivalence was somehow intuitively used in [10, 11], without any explicit and precise definition, and without proof, in the context of general exponential time algorithm (i.e., non parameterized). Even though this equivalence may look intuitive, proving this equivalence formally requires precise definitions and sophisticated proofs. As a matter of fact, this notion of equivalence was causing some confusion among researchers in parameterized complexity recently. We then use this notion to show that restricted instances of well-know parameterized NP-hard problems are as difficult as the general instances in terms of their parameterized subexponential time computability. In particular, we show that the Planar Dominating Set problem on degree-3 graphs (henceforth abbreviated Planar-3DS) can be solved in $O(2^{o(\sqrt{k})}p(n))$ (p is a polynomial) parameterized time if and only if the general Planar Dominating Set (abbreviated Planar-DS) problem can. Our results parallel the result in [11] for the Independent Set problem, in the context of the standard exponential time computability.

Apart from their complexity theoretic implications, our results also have an algorithmic flavor. For instance, in our proof of the above mentioned result we give a reduction from Planar-DS to Planar-3DS. This reduction shows that if Planar-3DS can be solved in time $O(2^{\sqrt{k}/3}n)$, then the Planar-DS problem can be solved in time $O(2^{15\sqrt{k}}n)$. Given that the currently most efficient algorithm for Planar-DS has running time $O(2^{15.13\sqrt{k}}n)$ [8], and that the structure of the Planar-3DS problem looks much simpler than that of Planar-DS, one could see a possibility of improving the algorithms for Planar-DS by working on Planar-3DS. For instance, Baker's layer-wise decomposition theorem for planar graphs has been used extensively in designing parameterized algorithms for Planar-DS. This decomposition theorem seems to have many nice properties that could be exploited when the graph has degree bounded by 3. Also, the layer-wise separators, heavily used in such algorithms as well, seems to have very special properties when the underlying graph has degree bounded by 3.

Throughout the paper, we assume basic familiarity with graphs and standard NP-hard problems. The reader is referred to [6, 9] for more details.

2 The equivalence theorem

In this section we will show that a parameterized problem Q admits an $f(k)$-uniform parameterized exponential time scheme if and only if Q can be solved in $O(2^{o(f(k))}q(n))$ parameterized time. Let Q be parameterized problem, and let $f(k)$ be a nondecreasing and unbounded function[1]. We will prove that the following two statements are equivalent:

(1) Q can be solved in time $O(2^{\delta f(k)}p(n))$ for any constant $\delta > 0$, where p is a polynomial;

(2) Q can be solved in time $O(2^{o(f(k))}q(n))$, where q is a polynomial.

It is clear that proving this equivalence will imply our statement above. We first give precise definitions for the above concepts.

Definition A parameterized problem Q is solvable in time $O(2^{\delta f(k)}p(n))$ (p is a polynomial) for any constant $\delta > 0$ if there exists a parameterized algorithm A for Q such that, on any given instance (x, k) of Q with $|x| = n$, and any constant $\delta > 0$, the running time of the algorithm A is bounded by $h_\delta(\delta)2^{\delta f(k)}p(n)$, where $h_\delta(\delta)$ is independent of k and n.

Remark 1. According to the above definition, the algorithm A runs in time $O(2^{\delta f(k)}p(n))$ for any fixed constant $\delta > 0$. However, we do not exclude the possibility that the constant hidden in the $O()$ notation also depends on δ. In particular, the function $h_\delta(\delta)$ can be "non-uniform" in the sense that different δ values may induce different functions h_δ hidden in the $O()$ notation. For example, for two different δ_1 and δ_2, we may have $h_{\delta_1}(\delta_1) = 2/\delta_1$, but $h_{\delta_2}(\delta_2) = 25^{1/\delta_2}$.

Remark 2. Besides the constant function h_δ, we only consider the "uniform" case in which we assume a single algorithm A for all constants $\delta > 0$. This convention has been used in the study of polynomial time approximation schemes, in which most proposed polynomial time approximation schemes are based on a single algorithm. Moreover, we also assume a uniform polynomial $p(n)$ for all constants $\delta > 0$. This also does not seem uncommon in the development of parameterized algorithms.

Definition A parameterized problem Q is solvable in time $O(2^{o(f(k))}p(n))$, where p is a polynomial, if there exists a nondecreasing unbounded function $r(k)$ such that the problem Q can be solved in time $O(2^{f(k)/r(k)}p(n))$, where the constant hidden in the $O()$ notation is independent of k and n.

Lemma 2.1 *Let $f(k)$ be a nondecreasing and unbounded function, and let Q be a parameterized problem solvable by an algorithm A_0 in time $h_\delta(\delta)2^{\delta f(k)}p(n)$ for all $\delta > 0$. Then if we let k_δ be the smallest integer such that $f(k_\delta) \geq 2\log h_{\delta/2}(\delta/2)/\delta$, then there is an algorithm A for Q such that, on any $\delta > 0$, the running time of A on an instance (x, k) of Q with $k \geq k_\delta$ is bounded by $2^{\delta f(k)}p(n)$, where $n = |x|$.*

[1] We only consider "nice" complexity functions. We always assume that all complexity functions, such as $f(k)$, are computable in time polynomial in n, and the function values are always larger than or equal to 1.

PROOF. Consider the following algorithm A. Given an instance (x,k) of Q and $\delta > 0$, the algorithm A simulates the algorithm A_0 on instance (x,k) and $\delta' = \delta/2$. The running time of A is bounded by $h_{\delta/2}(\delta/2)2^{\delta f(k)/2}p(n)$. Since $2^{\delta/2} > 1$ and $f(k)$ is nondecreasing and unbounded, for all $k \geq k_\delta$, we have $f(k) \geq 2\log h_{\delta/2}(\delta/2)/\delta$, which gives $2^{\delta f(k)/2} \geq h_{\delta/2}(\delta/2)$. Thus, on instances (x,k) of Q with $k \geq k_\delta$, the running time of the algorithm A is bounded by

$$h_{\delta/2}(\delta/2)2^{\delta f(k)/2}p(n) \leq 2^{\delta f(k)/2}2^{\delta f(k)/2}p(n) = 2^{\delta f(k)}p(n)$$

This completes the proof. □

Theorem 2.2 *Let $f(k)$ be a nondecreasing and unbounded function, and let Q be a parameterized problem. Then the following statements are equivalent:*

(1) *Q can be solved in time $O(2^{\delta f(k)}p(n))$ for any constant $\delta > 0$, where p is a polynomial;*
(2) *Q can be solved in time $O(2^{o(f(k))}q(n))$, where q is a polynomial.*

PROOF. Suppose that (2) holds and Q can be solved in time $O(2^{o(f(k))}q(n))$. By definition, there exists a nondecreasing and unbounded function $r(k)$ such that Q is solved by a parameterized algorithm A_1 whose running time is bounded by $c \cdot 2^{f(k)/r(k)}q(n)$, where c is a fixed constant independent of k and n. Since $f(k)$ and $r(k)$ are nondecreasing and unbounded, there must exist a $\bar{k}_\delta$ such that

$$f(k)/r(k) + \log c < \delta f(k)$$

for all $k \geq \bar{k}_\delta$. The value $\bar{k}_\delta$ depends only on δ.

Now consider the complexity of the algorithm A_1. For any instance (x,k) of Q and any $\delta > 0$, if $k < \bar{k}_\delta$, then the complexity of the algorithm A_1 is (note that $f(k)$ is nondecreasing):

$$c \cdot 2^{f(k)/r(k)}q(n) \leq c \cdot 2^{f(k)}q(n) \leq c \cdot 2^{f(\bar{k}_\delta)}q(n) = h_\delta(\delta)q(n) \leq h_\delta(\delta)2^{\delta f(k)}q(n)$$

where $h_\delta(\delta) = c \cdot 2^{f(\bar{k}_\delta)}$ is a function only depending on δ. On the other hand, if $k \geq \bar{k}_\delta$, then

$$c \cdot 2^{f(k)/r(k)}q(n) = 2^{f(k)/r(k)+\log c}q(n) < 2^{\delta f(k)}q(n) \leq h_\delta(\delta)2^{\delta f(k)}q(n)$$

Thus, the complexity of the algorithm A_1 is again bounded by $h_\delta(\delta)2^{\delta f(k)}q(n)$. This shows that if Q is solvable in time $O(2^{o(f(k))}q(n))$, then Q is also solvable in time $O(2^{\delta f(k)}q(n))$ for any $\delta > 0$.

Now suppose that (1) holds and Q is solvable in time $O(2^{\delta f(k)}p(n))$ for all $\delta > 0$. By definition, there exist a parameterized algorithm A for Q such that on any given instance (x,k) of Q and any constant $\delta > 0$, the running time of the algorithm A is bounded by $h_\delta(\delta)2^{\delta f(k)}p(|x|)$, where $h_\delta(\delta)$ is independent of k and n. By Lemma 2.1, there is another algorithm A_2 for Q such that, on any $\delta > 0$, the running time of A_2 on an instance (x,k) of Q with $k \geq k_\delta$ is bounded by $2^{\delta f(k)}p(n)$, where k_δ is defined to be the smallest integer satisfying $f(k_\delta) \geq 2\log h_{\delta/2}(\delta/2)/\delta$. Let $\delta_s = 1/s$, for $s = 1, 2, \ldots$. We define a sequence of integers

$$k_0 < k_1 < k_2 < \cdots < k_s < \cdots$$

as follows. Let $k_0 = 0$. Inductively, for each $i > 0$, we define $k_i = \max\{k_{i-1} + 1, k_{\delta_i}\}$, where k_{δ_i} is the smallest integer satisfying $f(k_{\delta_i}) \geq 2 \log h_{\delta_i/2}(\delta_i/2)/\delta_i$.

We first show, for a given k, how to compute in polynomial time the index t such that $k_t \leq k < k_{t+1}$. By definition, $k_0 = 0$. Inductively, suppose we have computed k_0, k_1, ..., k_{i-1} such that $k_{i-1} \leq k$. We then calculate $f(l)$, for $l = k_{i-1} + 1, k_{i-1} + 2, \ldots$, until either we reach $l = k$ or l satisfies $f(l) \geq 2 \log h_{\delta_i/2}(\delta_i/2)/\delta_i$. Note that in the latter case, l is exactly the value k_i. Therefore, after computing all values $f(1)$, $f(2)$, ..., $f(k)$, we should find the index t such that $k_t \leq k < k_{t+1}$. By our assumption, the function f is computable in polynomial time. Moreover, by our definition, $k_i \geq i$ for all i. Thus, $t \leq k$ for the index t satisfying $k_t \leq k < k_{t+1}$. So we only need to compute at most k values for $h_{\delta_i/2}(\delta_i/2)/\delta_i$, and again by our assumption, the value of $h_{\delta_i/2}(\delta_i/2)$ can be computed in polynomial time. In conclusion, for a given k, we can compute the index t such that $k_t \leq k < k_{t+1}$ in polynomial time.

Now we are ready for the proof. Construct an algorithm A_2' as follows. Given an instance (x, k) of the problem Q, the algorithm A_2' first computes the index t such that $k_t \leq k \leq k_{t+1}$, then simulates the algorithm A_2 on the instance (x, k) and δ_t. Since $k_t \leq k$ and $k_t \geq k_{\delta_t}$, the algorithm A_2 runs in time $2^{\delta_t f(k)} p(n) = 2^{f(k)/t} p(n)$. Thus, the running time of the algorithm A_2' is bounded by $2^{f(k)/t} p(n) + p_1(n)$ for $k_t \leq k < k_{t+1}$, where $p_1(n)$ is the polynomial time taken to compute the index t. For simplicity, we will ignore the polynomial term $p_1(n)$ in the following discussion.

Let $T(k, n)$ be the running time of the algorithm A_2'. From the above discussion, we have

$$T(n, k) \leq 2^{f(k)/t} p(n) \qquad \text{for } k_t \leq k < k_{t+1}$$

Let $F(k) = T(n, k)/p(n)$, then

$$F(k) \leq 2^{f(k)/t} \qquad \text{for } k_t \leq k < k_{t+1}$$

This gives

$$f(k)/\log F(k) \geq t \qquad \text{for } k_t \leq k < k_{t+1}$$

Now if we define

$$r(k) = t \qquad \text{for } k_t \leq k < k_{t+1}$$

then $r(k) \leq f(k)/\log F(k)$ for all k. Moreover, $r(k)$ is a nondecreasing and unbounded function.

From this, we can easily get $\log F(k) \leq f(k)/r(k)$, and $F(k) \leq 2^{f(k)/r(k)}$. By the definition of $F(k)$, we have $T(n, k)/p(n) \leq 2^{f(k)/r(k)}$. Finally

$$T(n, k) \leq 2^{f(k)/r(k)} p(n)$$

Since $T(k, n)$ is the running time of the algorithm A_2', and $r(k)$ is a nondecreasing and unbounded function, we conclude that the running time of the algorithm A_2' is $O(2^{o(f(k))} p(n))$. It follows that the problem Q can be solved in time $O(2^{o(f(k))} p(n))$. This completes the proof of the theorem. □

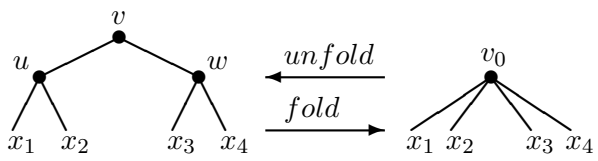

Figure 1: Vertex folding and unfolding

3 Hard instances of parameterized NP-hard problems

3.1 VC and VC-3

A set of vertices C is a *vertex cover* for G if every edge in G is incident to at least one vertex in C. In the parameterized VC problem (shortly the VC problem) we are given a pair (G, k) as input, where G is an undirected graph and k is a positive integer (the parameter), and we are asked to decide if G has a vertex cover of size bounded by k. The VC-3 problem is the set of instances of the VC problem in which the underlying graph has degree bounded by 3. For a graph G, denote by $\tau(G)$ the size of a minimum vertex cover of G. We will show in this section that the VC-3 problem is as hard as the general VC problem in terms of its parameterized subexponential time complexity. Let (G, k) be an instance of the VC problem. We will need the following propositions.

Proposition 3.1 [NT-Theorem] ([3, 15]). *There is an $O(\sqrt{n}m)$ time algorithm that, given a graph G of n vertices and m edges, constructs two disjoint subsets C_0 and V_0 of vertices in G such that*

(1) *Every minimum vertex cover of $G(V_0)$ plus C_0 forms a minimum vertex cover for G;*

(2) *A minimum vertex cover of $G(V_0)$ contains at least $|V_0|/2$ vertices.*

Proposition 3.1 allows us to assume, without loss of generality, that in an instance (G, k) of the VC problem, the graph G contains at most $2k$ vertices.

Let v be a degree-2 vertex in the graph G with two neighbors u and w such that u and w are not adjacent. We construct a new graph G' as follows: remove the vertices v, u, and w and introduce a new vertex v_0 adjacent to all neighbors of the vertices u and w in G (of course except the vertex v). We say that the graph G' is obtained from the graph G by *folding* the vertex v. See Figure 1 for an illustration of this operation.

Proposition 3.2 ([5]) *Let G' be a graph obtained by folding a degree-2 vertex v in a graph G, where the two neighbors of v are not adjacent to each other. Then $\tau(G) = \tau(G') + 1$. Moreover, a minimum vertex cover for G can be constructed from a minimum vertex cover for G' in linear time.*

We define an inverse operation of the folding operation that we call *unfold*. Given a vertex v_0 in a graph G where $d(v_0) > 3$, and with neighbors $x_1, \ldots, x_r$, we construct a graph G' as follows. Remove v_0 and introduce three new vertices v, u, and w. Connect v to u and w, connect u to x_1 and x_2, and connect w to $x_3, \ldots, x_r$ (see Figure 1).

From Proposition 1, we know that $\tau(G') = \tau(G) + 1$. Moreover, the unfold(v_0) operation replaces v_0 with three new vertices: v of degree 2, u of degree 3, and w of degree $d(v_0) - 1$. Now if

$d(w) > 3$, we can apply the unfold(w) operation, and so on, until all the newly introduced vertices have degree bounded by 3. It is easy to check that exactly $d(v_0) - 3$ operations are needed to replace v_0 by new vertices each having a degree bounded by 3. Let us call this iterative process initiated at the vertex v_0 *iterative-unfold(v_0)*. If G'' is the resulting graph from G after applying *iterative-unfold(v_0)*, then from the above discussion we have $\tau(G'') = \tau(G) + d(v_0) - 3$. Since each unfold() operation increases the number of vertices in the graph by 2, the number of vertices n'' in G'' is $n + 2d(v_0) - 6$, where n is the number of vertices in G.

Theorem 3.3 *The* VC-3 *problem can be solved in $O(2^{o(k)}p(n))$ time if and only if the* VC *problem can be solved in $O(2^{o(k)}q(n))$ time, where n is the number of vertices in the graph, and p, q are two polynomials.*

PROOF. Obviously, if VC can be solved in $O(2^{o(k)}q(n))$ time then so can VC-3. To prove the other direction, suppose that VC-3 can be solved in $O(2^{o(k)}p(n))$ time for some polynomial p. By Theorem 2.2, VC-3 can be solved in time $O(2^{\epsilon k}p(n))$ for any $0 < \epsilon < 1$. To show that VC can be solved in time $O(2^{o(k)}q(n))$, by Theorem 2.2, it suffices to show that it can be solved in $O(2^{\delta k}q(n))$ time (q is a polynomial) for any $0 < \delta < 1$. Let (G, k) be an instance of the VC problem, and let $0 < \delta < 1$ be given. Consider the scheme in Figure 2.

VC-scheme

Input: an instance (G, k) of VC and a constant $0 < \delta < 1$
Output: a vertex cover C of G of size bounded by k in case it exists

1. apply Proposition 3.1 to G; (without loss of generality, assume the resulting instance is (G, k))
2. let $d = \lceil -2\lg(2^{\delta/2} - 1)/\delta \rceil$;
3. **while** there exists a vertex v of degree $> d$ **do** branch as v and $N(v)$;
4. **while** there exists a vertex v of degree > 3 **do** **iterative-unfold(v)**;
5. call the VC-3 scheme on the resulting graph and parameter,
   with $\epsilon = \delta/(4d - 10)$, to compute the desired vertex cover C' in case it exists;
6. **if** C' exists **then** using Proposition 3.2 and C' output the desired vertex cover C of G;

Figure 2: A scheme for VC

We are implicitly assuming that at each step the above algorithm, the graph and the parameter are updated appropriately. Keeping this in mind, it is not difficult to see the correctness of the above algorithm. The only step that may need additional explanation is step 3. Basically, for any vertex v in G, it is easy to see that either there exists a minimum vertex cover containing v, or a minimum vertex cover containing its set of neighbors $N(v)$. Thus, the branch in step 3 is correct. Also note that at the end of step 4 every vertex in the resulting graph has degree bounded by 3. The correctness of the other steps follows from Proposition 3.1 and Proposition 3.2.

We analyze the running time of the algorithm. By proposition 3.1, step 1 takes polynomial time in n, and the resulting parameter is not larger than the initial parameter k. In step 3 we branch according to the recurrence relation $T(k) \leq T(k - d - 1) + T(k - 1)$, which has a solution $T(k) = O(r^k)$, where r is the unique root of the polynomial $p(x) = x^{d+1} - x^d - 1$ in the interval $[1, 2]$

(see [5]). It is easy to verify that, with the choice of d in step 2, this recurrence relation has solution $O(2^{\delta k/2})$ [5]. In step 4 we apply the subroutine **iterative-unfold** to every vertex of degree > 3. For every vertex v in the graph of degree > 3, **iterative-unfold**(v) can increase the parameter by no more than $d(v) - 3 \leq d - 3$, and the number of vertices in the graph by no more than $2(d-3)$, since at this point of the algorithm the degree of the graph is bounded by d (note that $d > 3$). By Proposition 3.1, the number of vertices in the graph is bounded by $2k$, and hence, after step 4, the new parameter k' is bounded by $k + 2k(d-3) = (2d-5)k$, and the number of vertices n' is bounded by $2k + 4k(d-3) = (4d-10)k$. Clearly, the running time of step 4 is polynomial. In step 5, the **VC-3** scheme is called with $\epsilon = \delta/(4d-10)$. By our assumption, the **VC-3** scheme runs in time $O(2^{\epsilon k'} p(n))$. It follows that step 5 takes time $O(2^{\delta(2d-5)k/(4d-10)} p(n)) = O(2^{\delta k/2} p(n))$, and the total running time of the algorithm is $O(2^{\delta k/2}.2^{\delta k/2} q(n)) = O(2^{\delta k} q(n))$, for some polynomial q. The theorem follows. □

3.2 Planar-DS and Planar-3DS

A *dominating set* D in G is a set of vertices such that every vertex in G is either in D or adjacent to a vertex in D. The parameterized Planar-DS problem takes as input a pair (G, k), where G is a planar graph, and asks to decide if G has a dominating set of size bounded by k. The Planar-3DS problem is the Planar-DS problem restricted to graphs of degree bounded by 3. For a graph G, denote by $\eta(G)$ the size of a minimum dominating set in G. Let (G, k) be an instance of the Planar-DS problem. We will need the following propositions.

Proposition 3.4 *([1]). There is an $O(n^3)$ time algorithm that, given an instance (G, k) of* Planar-DS*, where G has n vertices, produces an instance (G', k') of* Planar-DS*, where G' has n' vertices, such that: (1) $n' \leq n$ and $k' \leq k$; (2) $n' \leq 335k'$; (3) G' has a dominating set of size $\leq k'$ if and only if G has a dominating set of size $\leq k$; and (4) from a solution D' of G' a solution D of G can be constructed in linear time.*

By Proposition 3.4, we can assume that in an instance (G, k) of the Planar-DS problem, the graph G contains at most $335k$ vertices.

Let v be a vertex in the graph G of degree > 3 and let $w_1, \ldots, w_r$, $r > 3$, be the neighbors of v. We construct a new graph G' from G as follows. Remove v and introduce four new vertices x, x', y, y'. Connect x to w_1 and w_2, y to $w_3, \ldots, w_r$, x' to x, y' to y, and x' to y'. We say that the graph G' is obtained from the graph G by *expanding* the vertex v. See Figure 3 for an illustration of this operation. It is clear that this operation preserves the planarity of the graph.

Proposition 3.5 *Let G' be a graph obtained by expanding a vertex v of degree > 3 in a planar graph G. Then $\eta(G) = \eta(G') - 1$. Moreover, a minimum dominating set for G can be constructed from a minimum dominating set for G' in linear time.*

Proof. We first show that $\eta(G') \leq \eta(G) + 1$. Let D be a minimum dominating set for G. If D contains v, then clearly $(D - \{v\}) \cup \{x, y\}$ is a dominating set for G' of size $\eta(G) + 1$. If D does not contain v, then D must contain at least one vertex in $\{w_1, \ldots, w_r\}$ (since v must be dominated). If

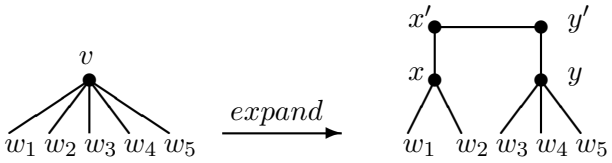

Figure 3: Vertex expansion

D contains a vertex in $\{w_1, w_2\}$ then $D \cup \{y'\}$ is a dominating set for G' of size $\eta(G)+1$, whereas if D contains a vertex in $\{w_3, \ldots, w_r\}$, then $D \cup \{x'\}$ is a dominating set for G' of size $\eta(G)+1$. It follows that in all cases G' has a dominating set of size $\eta(G)+1$, and hence $\eta(G') \leq \eta(G)+1$. Now to prove that $\eta(G) \leq \eta(G')-1$, let D' be a minimum dominating set for G'. We distinguish the following cases.

Case 1. D' contains both x and y. In this case $(D' - \{x, y, x', y'\}) \cup \{v\}$ is a dominating set for G of size bounded by $\eta(G')-1$.

Case 2. D' contains exactly one vertex in $\{x, y\}$, without loss of generality, let this vertex be x (the other case is symmetrical). Then D' must contain at least one vertex in $\{x', y'\}$. Thus, $(D' - \{x, x', y'\}) \cup \{v\}$ is a dominating set for G of size bounded by $|D| - 1 = \eta(G') - 1$.

Case 3. D' does not contain any vertex in $\{x, y\}$, then D' has to contain at least one vertex in $\{x', y'\}$. If D' contains at least one vertex in $\{w_1, \ldots, w_r\}$, then $D' - \{x', y'\}$ is a dominating set for G of size bounded by $\eta(G')-1$. On the other hand if D' does not contain any vertex in $\{w_1, \ldots, w_r\}$, then D' must contain both x' and y' in order to cover x and y. Now $(D' - \{x', y'\}) \cup \{v\}$ is a dominating set for G' of size $\eta(G')-1$.

Thus, in all cases G has a dominating set of size bounded by $\eta(G')-1$. It follows that $\eta(G) \leq \eta(G')-1$, and hence, $\eta(G') = \eta(G)+1$. Moreover, given a dominating set D' of G', it should be clear how the corresponding dominating set D of G can be constructed in linear time according to one of the above three cases. □

If v is a vertex in G such that $d(v) > 3$, the operation expand(v) replaces v with four new vertices: x of degree 3, x' of degree 2, y' of degree 2, and y of degree $d(v)-1$. If $d(y) > 3$, we can apply the expand(y) operation, and so on, until all the newly introduced vertices have degree bounded by 3. Again exactly $d(v)-3$ operations are needed to replace v by new vertices each having a degree bounded by 3. We denote this iterative process initiated at the vertex v *iterative-expand(v)*. If G'' is the resulting graph from G after applying *iterative-expand(v)*, then we have $\eta(G'') = \eta(G) + d(v) - 3$, and the number of vertices n'' of G'' is $n + 3d(v) - 9$.

Theorem 3.6 *The* Planar-3DS *problem can be solved in $O(2^{o(\sqrt{k})}p(n))$ time if and only if the* Planar-DS *problem can be solved in $O(2^{o(\sqrt{k})}q(n))$ time, where n is the number of vertices in the graph, and p, q are two polynomials.*

Proof. The proof of this theorem has the same flavor as Theorem 3.3. First if Planar-DS can

be solved in $O(2^{o(\sqrt{k})}q(n))$ time then so can PLANAR-3DS. To prove the other direction, we suppose that PLANAR-3DS can be solved in time $O(2^{\epsilon\sqrt{k}}p(n))$ for any $0 < \epsilon < 1$, and for some polynomial p, and we show that PLANAR-DS can be solved in $O(2^{\delta\sqrt{k}}q(n))$ time (q is a polynomial) for any $0 < \delta < 1$. By Theorem 2.2, this will be sufficient. Let (G, k) be an instance of the PLANAR-DS problem, and let $0 < \delta < 1$ be given. Consider the scheme in Figure 4.

DS-scheme

Input: an instance (G, k) of PLANAR-DS and a constant $0 < \delta < 1$
Output: a dominating set D of G of size bounded by k in case it exists

1. apply Proposition 3.4 to G;
2. **while** there exists a vertex v of degree > 3 **do** **iterative-expand**(v);
3. call the PLANAR-3DS scheme on the resulting graph and parameter, with $\epsilon = \delta/\sqrt{2011}$, to compute the desired dominating set D' in case it exists;
4. **if** D' exists **then** using Proposition 3.5 and D' output the desired dominating set D of G;

Figure 4: A scheme for Planar-DS

The analysis of the algorithm and its correctness follows a similar line to that of Theorem 3.3. However, few things need to be clarified. First, after step 1, we know by Proposition 3.4 that the number of vertices n in G is bounded by $335k$. In step 2, the **iterative-expand()** operation increases both the parameter and the number of vertices in G. Let G' be the resulting graph at the end of step 2, and let k' and n' be the parameter and number of vertices in G', respectively. Each call to **iterative-expand**(v), where $d(v) > 3$, increases k by $d(v) - 3$ and n by $3d(v) - 9$. It follows that

$$\begin{aligned} k' &= k + \sum_{v \in G, d(v) > 3} (d(v) - 3) \leq k + \sum_{v \in G} d(v) \\ &\leq k + 6n - 12 < k + 2010k = 2011k \end{aligned}$$

The last two inequalities follow from the fact that the number of edges in a planar graph of n vertices is bounded by $3n - 6$ [6], and from Proposition 3.4. Similarly, we can show that $n' \leq 19n$. It is easy now to see that the theorem follows. □

3.3 MAX-SAT and MAX-CUT

The parameterized MAX-SAT problem is defined as follows. Given a boolean formula F in conjunctive normal form and a positive integer k, decide if F has a truth assignment that satisfies k or more clauses. The parameterized MAX-3SAT problem is the parameterized MAX-SAT problem restricted to formulas in which each clause contains at most three literals. Mahajan and Raman [14] showed the following.

Proposition 3.7 ([14]) *Given a formula F and a positive integer k, then in linear time, we can compute a formula G and a positive integer $k' \leq k$ with $|G| \in O(k'^2)$, such that F has an assignment satisfying at least k clauses if and only if G has an assignment satisfying at least k'*

clauses. Moreover, such an assignment for F is computable from an assignment for G in linear time.

Using Proposition 3.7, the standard reduction from SAT to 3-SAT [9], and Theorem 2.2, we can show the following theorem.

Theorem 3.8 *If parameterized* Max-3SAT *can be solved in time $O(2^{o(\sqrt{k})}p(n))$ then parameterized* Max-SAT *can be solved in time $O(2^{o(k)}q(n))$, where p and q are two polynomials.*

In the parameterized Max-Cut problem we are given an undirected graph G and a positive integer k, and we are asked to decide if the vertex set of G can be partitioned into two parts so that at least k edges cross the partitioning. The parameterized Max-3Cut problem is the parameterized Max-Cut problem on graphs of degree bounded by 3. We have the following theorem.

Theorem 3.9 *Parameterized* Max-3Cut *can be solved in time $O(2^{o(k)}p(n))$ if and only if parameterized* Max-Cut *can be solved in time $O(2^{o(k)}q(n))$, where p and q are two polynomials.*

4 Concluding remarks

In this paper we showed the equivalence of the two notions: an algorithm having an $f(k)$-uniform parameterized exponential time scheme and an algorithm being solvable in $O(2^{o(f(k))}p(n))$ parameterized time. We used this equivalence to identify some hard special instances of well-known NP-hard parameterized problems like Vertex Cover, Planar-DS, Max-3SAT, and Max-3Cut. Our results can have algorithmic implications in addition to their complexity theoretic ones.

A final remark that is worth mentioning is the following. Our results show that 3-VC, Planar-3DS, and Max-3Cut, are "as hard" as the corresponding general problems. These results do not seem to generalize in the same manner to Max-SAT, as we have seen in the previous section. The problem that arises with Max-SAT is very similar to that that arises when trying to show the same result for 3-SAT and SAT, with respect to the standard exponential time computability framework.

The question whether 3-SAT and SAT have subexponential time algorithms seems to have direct consequences on parameterized complexity: 3-SAT in DTIME($2^{o(n)}$) would imply that Vertex Cover, for instance, can be solved in parameterized subexponential time by our results and those in [4], and SAT in DTIME($2^{o(n)}$) would imply that W[1] = FPT [7]. It would be interesting to study what elements contribute to the differentiation between the time complexity of Max-3SAT and Max-SAT, and 3-SAT and SAT. It seems one of the major differences between such problems and problems like Vertex Cover, is that in SAT and Max-SAT there are two parameters that affect the complexity, namely the number of variables and the number of clauses. This suggests that there might be some threshold that is somehow related to the "density" of the formula (total number of occurrences of literals divided by the total number of clauses) that is affecting the (parameterized) exponential time computability of these problems. This difference and the existence of a threshold are worth investigation.

References

[1] J. Alber, M. Fellows, and R. Niedermeier, Efficient Data Reduction for Dominating Set: A Linear Problem Kernel for the Planar Case, *Lecture Notes in Computer Science (LNCS)* **2368**, (2002), pp. 150-159.

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

[3] R. Bar-Yehuda and S. Even, A local-ratio theorem for approximating the weighted vertex cover problem, *Annals of Discrete Mathematics* **25**, (1985), pp. 27-46.

[4] L. Cai and D. Juedes, On the existence of subexponential-time parameterized algorithms, available at `http://www.cs.uga.edu/ ~ cai/`.

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

[6] R. Diestel, *Graph Theory*, New York, New York: Springer, (1996).

[7] R. Downey and M. Fellows, *Parameterized Complexity*, New York, New York: Springer, (1999).

[8] F. Fomin and D. Thilikos, Dominating sets in planar graphs: branch-width and exponential speed-up, *Proc. 14th ACM-SIAM Symp. Discrete Algorithms* (SODA'03), (2003), pp. 168-177.

[9] M. Garey and D. Johnson, *Computers and Intractability: A Guide to the Theory of NP-completeness*, Freeman, San Francisco, 1979.

[10] R. Impagliazzo, R. Paturi, and F. Zane, Which Problems Have Strongly Exponential Complexity?, *Journal of Computer and System Sciences (JCSS)* **63-4**, (2001), pp. 512-530.

[11] D. Johnson and M. Szegedy, What are the least tractable instances of max. independent set?, *Proceedings of the (SODA'99)*, (1999), pp. 927-928.

[12] D. Juedes, Parameterized lower bounds: new results and new directions, *International Workshop on Complexity and Parameters: Logic and Structures*, Kanpur, December 2002.

[13] I. Kanj and L. Perkovic, Improved parameterized algorithms for planar dominating set, *Lecture Notes in Computer Science* **2420** (MFCS'02), (2002), pp. 399-410.

[14] M. Mahajan and V. Raman, Parameterizing above guaranteed values: Max-Sat and Max-Cut, *Journal of Algorithms* **31**, (1999), pp. 335-354.

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