On PTAS for Planar Graph Problems

Xiuzhen Huang[1] and Jianer Chen[2]

[1] Department of Computer Science,
Arkansas State University,
State University, Arkansas 72467.
Email: xzhuang@csm.astate.edu
[2] Department of Computer Science,
Texas A&M University,
College Station, TX 77843.
Email: chen@cs.tamu.edu**

Abstract. Approximation algorithms for a class of planar graph problems, including PLANAR INDEPENDENT SET, PLANAR VERTEX COVER and PLANAR DOMINATING SET, were intensively studied. The current upper bound on the running time of the polynomial time approximation schemes (PTAS) for these planar graph problems is of $2^{O(1/\epsilon)}n^{O(1)}$. Here we study the lower bound on the running time of the PTAS for these planar graph problems. We prove that there is no PTAS of time $2^{o(\sqrt{1/\epsilon})}n^{O(1)}$ for PLANAR INDEPENDENT SET, PLANAR VERTEX COVER and PLANAR DOMINATING SET unless an unlikely collapse occurs in parameterized complexity theory. For the gap between our lower bound and the current known upper bound, we specifically show that to further improve the upper bound on the running time of the PTAS for PLANAR VERTEX COVER, we can concentrate on PLANAR VERTEX COVER on planar graphs of degree bounded by three.

1 Introduction

There is intensive research work on a class of planar graph NP-hard optimization problems, such as PLANAR INDEPENDENT SET, PLANAR VERTEX COVER and PLANAR DOMINATING SET. Approximation algorithms for these planar graph problems and related problems were studied by researchers such as Bar-Yehuda and Even [5], Lipton and Tarjan [25], Baker [4], Eppstein [16], Grohe [20], Khanna and Motiwani [24], and Cai et al. [7]. The current upper bound on the running time of the polynomial time approximation scheme (PTAS) for these planar graph problems is of $2^{O(1/\epsilon)}n^{O(1)}$ [4, 25]. In this paper, we study the lower bound on the running time of the PTAS algorithms for these planar graph problems. Our work follows some recent research progress in parameterized complexity theory [10, 11], where strong computational lower bound results on the running time of the algorithms for $W[t]$-hard problems are derived, $t \geq 1$.

** This research is supported in part by US NSF under Grants CCR-0311590 and CCF-0430683.

Our research work here is focused on the computational lower bounds on the running time of the algorithms for the parameterized problems that are fixed-parameter tractable (in FPT).

We first give a brief review on parameterized complexity theory and the recent research results in [10, 11]. A *parameterized problem* Q is a decision problem consisting of instances of the form (x, k), where the integer $k \geq 0$ is called the *parameter*. The parameterized problem Q is *fixed-parameter tractable* [15] if it can be solved in time $f(k)|x|^{O(1)}$, where f is a recursive function[3]. Certain NP-hard parameterized problems, such as VERTEX COVER, are fixed-parameter tractable, and hence can be solved practically for small parameter values [12]. On the other hand, the inherent computational difficulty for solving many other NP-hard parameterized problems with even small parameter values has suggested that certain parameterized problems are not fixed-parameter tractable, which has motivated the theory of *fixed-parameter intractability* [15]. The W-hierarchy $\bigcup_{t \geq 0} W[t]$ has been introduced to characterize the inherent level of intractability for parameterized problems. A large number of parameterized problems have been proved to be hard or complete for various levels in the W-hierarchy [15]. Examples of $W[1]$-hard problems include many well-known NP-hard problems such as CLIQUE, DOMINATING SET, SET COVER, and WEIGHTED CNF SATISFIABILITY. The theory of parameterized intractability has found important applications in a variety of areas such as database systems and model checking [20, 27].

The $W[1]$-hardness of a parameterized problem provides a strong evidence that the problem is not fixed-parameter tractable, or equivalently, cannot be solved in time $f(k)n^{O(1)}$ for any function f. Recent investigation has derived much stronger computational lower bounds on the running time of the algorithms for well-known NP-hard parameterized problems [10, 11]. For example, it has been shown that unless an unlikely collapse occurs in the parameterized complexity theory, any algorithm solving the $W[1]$-hard CLIQUE problem takes time at least $n^{\Omega(k)}$. Note that this lower bound is asymptotically tight in the sense that the trivial algorithm that enumerates all subsets of k vertices in a given graph to test the existence of a clique of size k runs in time $O(n^k)$. Similar lower bound results could be shown for other $W[t]$-hard problems, $t \geq 1$.

A method for deriving lower bounds on the running time of approximation algorithms for NP-hard combinatorial optimization problems is designed. It was proved in [11] that unless an unlikely collapse occurs in parameterized complexity theory, the $W[1]$-hardness of the parameterized problem under the linear fpt-reduction implies the nonexistence of polynomial time approximation schemes of running time $f(1/\epsilon)n^{o(1/\epsilon)}$ for the original optimization problem, where f is *any* recursive function.

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

2 Terminologies in Approximation

For a reference of the theory of approximation, the readers are referred to the book [3]. In this section, we provide some basic terminologies for studying approximability and its relationship with parameterized complexity.

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

1. I_Q is the set of input instances. It is recognizable in polynomial time;

2. For each instance $x \in I_Q$, $S_Q(x)$ is the set of feasible solutions for x, which is defined by a polynomial p and a polynomial time computable predicate π (p and π only depend on Q) as $S_Q(x) = \{y : |y| \leq p(|x|) \text{ and } \pi(x,y)\}$;

3. $f_Q(x,y)$ is the objective function mapping a pair $x \in I_Q$ and $y \in S_Q(x)$ to a non-negative integer. The function f_Q is computable in polynomial time;

4. $opt_Q \in \{\max, \min\}$. Q is called a *maximization problem* if $opt_Q = \max$, and a *minimization problem* if $opt_Q = \min$.

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

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

$$opt_Q(x)/f_Q(x, y_A(x)) \leq r(|x|) \text{ if } Q \text{ is a maximization problem}$$
$$f_Q(x, y_A(x))/opt_Q(x) \leq r(|x|) \text{ if } Q \text{ is a minimization problem}$$

The approximation algorithm A has an *approximation ratio* $r(n)$ if for any instance x in I_Q, the solution $y_A(x)$ constructed by the algorithm A has an approximation ratio bounded by $r(|x|)$.

Definition 1. *An NP optimization problem Q has a polynomial-time approximation scheme (PTAS) if there is an algorithm A_Q that takes a pair (x, ϵ) as input, where x is an instance of Q and $\epsilon > 0$ is a real number, and returns a feasible solution y for x such that the approximation ratio of the solution y is bounded by $1 + \epsilon$, and for each fixed $\epsilon > 0$, the running time of the algorithm A_Q is bounded by a polynomial of $|x|$.*[4]

An NP optimization problem Q has a *fully polynomial-time approximation scheme* (FPTAS) if it has a PTAS A_Q such that the running time of A_Q is bounded by a polynomial of $|x|$ and $1/\epsilon$.

[4] There is an alternative definition for PTAS in which each $\epsilon > 0$ may correspond to a different approximation algorithm A_ϵ for Q [19]. The definition we adopt here may be called the *uniform PTAS*, by which a single approximation algorithm takes care of all values of ϵ. Note that most PTAS developed in the literature are uniform PTAS.

Observe that the time complexity of a PTAS algorithm may be of the form $O(2^{1/\epsilon}|x|^c)$ for a fixed constant c or of the form $O(|x|^{1/\epsilon})$. Obviously, the latter type of computations with small ϵ values will turn out to be practically infeasible. This leads to the following definition [9].

Definition 2. *An NP optimization problem Q has an efficient polynomial-time approximation scheme (EPTAS) if it admits a polynomial-time approximation scheme whose time complexity is bounded by $O(f(1/\epsilon)|x|^c)$, where f is a recursive function and c is a constant.*

An NP optimization problem Q can be parameterized in a natural way as follows.

Definition 3. *Let $Q = (I_Q, S_Q, f_Q, opt_Q)$ be an NP optimization problem. The parameterized version of Q is defined as follows:*

(1) If Q is a maximization problem, then the parameterized version of Q is defined as $Q_{\geq} = \{(x, k) \mid x \in I_Q \wedge opt_Q(x) \geq k\}$;

(2) If Q is a minimization problem, then the parameterized version of Q is defined as $Q_{\leq} = \{(x, k) \mid x \in I_Q \wedge opt_Q(x) \leq k\}$.

The above definition offers the possibility to study the relationship between the approximability and the parameterized complexity of NP optimization problems. However, there is an essential difference between the two categories: an approximation algorithm for an NP optimization problem constructs a solution for a given instance of the problem, while a parameterized algorithm only provides a "yes/no" decision on an input. To make the comparison meaningful, we need to extend the definition of parameterized algorithms in a natural way so that when a parameterized algorithm returns a "yes" decision, it also provides an "evidence" to support the conclusion (see [6] for a similar treatment).

Definition 4. *Let $Q = (I_Q, S_Q, f_Q, opt_Q)$ be an NP optimization problem. We say that a parameterized algorithm A_Q solves the parameterized version of Q if*

(1) in case Q is a maximization problem, then on an input pair (x, k) in $Q_{\geq}$, the algorithm A_Q returns "yes" with a solution y in $S_Q(x)$ such that $f_Q(x, y) \geq k$, and on any input not in $Q_{\geq}$, the algorithm A_Q simply returns "no";

(2) in case Q is a minimization problem, then on an input pair (x, k) in $Q_{\leq}$, the algorithm A_Q returns "yes" with a solution y in $S_Q(x)$ such that $f_Q(x, y) \leq k$, and on any input not in $Q_{\leq}$, the algorithm A_Q simply returns "no".

3 Lower Bound on Running Time of PTAS for Planar Graph Problems

Suppose $\epsilon > 0$ is the given error bound, and n is the number of vertices of a planar graph. Lipton and Tarjan [25] designed an EPTAS approximation algorithm of time $O(2^{O(1/\epsilon)}n^{O(1)})$ for PLANAR INDEPENDENT SET, as an application of a separator theorem on planar graphs. Based on the outer-planarity of planar graphs, Baker [4] designed EPTAS algorithms of time $O(2^{O(1/\epsilon)}n)$ for several famous NP-hard optimization problems on planar graphs, such as PLANAR VERTEX COVER, PLANAR INDEPENDENT SET, and PLANAR DOMINATING SET.

In [6], Cai and Chen proved that if an optimization problem has a fully polynomial-time approximation scheme (FPTAS), then the corresponding parameterized problem is fixed-parameter tractable (in FPT). Later this result was extended in [9] by Cesati and Trevisan: All optimization problems that have efficient polynomial time approximation schemes (EPTAS) have their parameterized problems in FPT. Therefore, the parameterized versions of these aforementioned optimization problems, PLANAR VERTEX COVER, PLANAR INDEPENDENT SET, and PLANAR DOMINATING SET, are in FPT.

Alber et. al [2] designed parameterized algorithms of time $2^{O(\sqrt{k})}n^{O(1)}$ for the parameterized versions of the above NP-hard optimization problems. A lot of research has been done on these problems to try to further improve the time complexity of the parameterized algorithms. Interested readers are referred to [1, 23, 17, 18].

Cai et. al [8] proved the following lower bound result for the parameterized algorithms of these problems:

Lemma 1. *(Lemma 5.1 in [8])* PLANAR VERTEX COVER*,* PLANAR INDEPENDENT SET*, and* PLANAR DOMINATING SET *do not have parameterized algorithms of time* $2^{o(\sqrt{k})}n^{O(1)}$*, unless* VERTEX COVER-3 *has* $2^{o(k)}n^{O(1)}$*-time parameterized algorithms.*

The class SNP introduced by Papadimitriou and Yannakakis [26] contains many well-known NP-hard problems including, for any fixed $q \geq 3$, CNF q-SAT, q-COLORABILITY, q-SET COVER, and VERTEX COVER, CLIQUE, and INDEPENDENT SET [22]. It is commonly believed that it is unlikely that all problems in SNP are solvable in subexponential time. Impagliazzo, Paturi and Zane [22] studied the class SNP and identified a group of SNP-complete problems under the serf-reduction, such that if any of these SNP-complete problems is solvable in subexponential time, then all problems in SNP are solvable in subexponential time. This group of SNP-complete problems under the serf-reduction includes the problems CNF q-SAT, q-COLORABILITY, q-SET COVER, and VERTEX COVER, CLIQUE, and INDEPENDENT SET.

We have:

Lemma 2. *(Theorem 3.3 in [13]) The* VERTEX COVER-3 *problem can be solved in* $2^{o(k)}n^{O(1)}$ *time if and only if the* VERTEX COVER *problem can be solved in* $2^{o(k)}n^{O(1)}$ *time.*

Therefore Lemma 1 could be restate as:

Lemma 3. PLANAR VERTEX COVER, PLANAR INDEPENDENT SET, *and* PLANAR DOMINATING SET *do not have parameterized algorithms of time* $2^{o(\sqrt{k})}n^{O(1)}$, *unless all SNP problems are solvable in subexponential time.*

We prove the following lower bound results on the running time of the EPTAS algorithms for those planar graph problems:

Theorem 1. PLANAR VERTEX COVER, PLANAR INDEPENDENT SET, *and* PLANAR DOMINATING SET *have no EPTAS of running time* $2^{o(\sqrt{1/\epsilon})}n^{O(1)}$, *where* $\epsilon > 0$ *is the given error bound, unless all SNP problems are solvable in subexponential time.*

Proof. We provide the proof for PLANAR VERTEX COVER. Let Q be the minimization problem of PLANAR VERTEX COVER.

From the EPTAS algorithm A_Q for the PLANAR VERTEX COVER problem Q, we provide the parameterized algorithm $A_\leq$ shown in Fig. 1 for the parameterized version $Q_\leq$ of the PLANAR VERTEX COVER problem Q.

Algorithm $A_\leq$**:**

Input: An instance (G, k) of $Q_\leq$, where G is a planar graph.

Output: If the minimum vertex cover C_0 has the size $|C_0| \leq k$, then Output "yes"; otherwise Output "no".

1. On the instance (G, k) of $Q_\leq$, **call** the EPTAS algorithm A_Q on G and $\epsilon = 1/(2k+1)$. Suppose that the algorithm A_Q returns a vertex cover C.

2. **If** $|C| \leq k$, then **return** "yes"; otherwise **return** "no".

Fig. 1. Algorithm $A_\leq$.

We verify that the algorithm $A_\leq$ solves the parameterized problem $Q_\leq$. Since the PLANAR VERTEX COVER problem Q is a minimization problem, if $|C| \leq k$ then obviously $|C_0| \leq k$. Thus, the algorithm $A_\leq$ returns a correct decision in this case. On the other hand, suppose $|C| > k$. Since $|C|$ is an integer, we have $|C| \geq k+1$. Since A_Q is a EPTAS for the PLANAR VERTEX COVER problem Q and $\epsilon = 1/(2k+1)$, we must have

$$|C|/|C_0| \leq 1 + 1/(2k+1)$$

From this we get (note that $|C| \geq k+1$)

$$|C_0| \geq |C|/(1 + 1/(2k+1) \geq (k+1)/(1 + 1/(2k+1) = k + 1/2 > k$$

Thus, in this case the algorithm $A_\leq$ also returns a correct decision. This proves that the algorithm $A_\leq$ solves the parameterized version $Q_\leq$ of the PLANAR

VERTEX COVER problem Q. The running time of the algorithm $A_{\leq}$ is dominated by that of the algorithm A_Q, which is bounded by $2^{o(\sqrt{1/\epsilon})}n^{O(1)} = 2^{o(\sqrt{k})}n^{O(1)}$. Thus, the parameterized version $Q_{\leq}$ of the PLANAR VERTEX COVER problem is solvable in time $2^{o(\sqrt{k})}n^{O(1)}$. Therefore, the result in the theorem follows from Lemma 3.

The proofs for PLANAR INDEPENDENT SET and PLANAR DOMINATING SET are similar and hence are omitted.

Corollary 1. PLANAR VERTEX COVER*,* PLANAR INDEPENDENT SET*, and* PLANAR DOMINATING SET *have no PTAS of running time* $2^{o(\sqrt{1/\epsilon})}n^{O(1)}$*, where* $\epsilon > 0$ *is the given error bound, unless all SNP problems are solvable in subexponential time.*

By a comparison with the upper bound on the running time of the EPTAS algorithms for these planar graph problems in Baker [4], which is $2^{O(1/\epsilon)}n^{O(1)}$ (also in Lipton and Tarjan [25]), we can see that there is a gap between the upper bound result and our lower bound result in Theorem 1. To come up with new approaches to improve the upper bound on the running time of the EPTAS algorithms in [4] will be interesting research. To study this issue, we concentrate on the PLANAR VERTEX COVER problem in the next section.

4 Upper Bound on Running Time of PTAS for Planar Vertex Cover

In this section, we study the PTAS algorithms for the VERTEX COVER problem on planar graphs of degree bounded by 3, abbreviated as P-VC-3. The VERTEX COVER problem on general planar graphs is abbreviated as P-VC.

From the proof of Theorem 1, we get the following lemma:

Lemma 4. *The* P-VC-3 *problem has no EPTAS of running time* $2^{o(\sqrt{1/\epsilon})}n^{O(1)}$*, where* $\epsilon > 0$ *is the given error bound, unless the* P-VC-3 *problem has a parameterized algorithm of time* $2^{o(\sqrt{k})}n^{O(1)}$*.*

It is well known that a planar embedding of a planar graph can be constructed in linear time [21]. We define an operation, called the *unfolding operation*, based on a planar embedding of a planar graph.

Definition 5. *Suppose that* G *is a planar graph with a planar embedding* $\pi(G)$*, and that* v *is a degree-*d *vertex in* G*, where* $d > 3$*, with neighbors* $v_1, v_2, \ldots, v_d$*, such that when one traverses around the vertex* v *on the embedding* $\pi(G)$*, the edges incident to* v *are in the cyclic order* $[v, v_1], [v, v_2], \ldots, [v, v_d]$*. The unfolding operation on the vertex* v *will do the following: remove the vertex* v *from* $\pi(G)$*, and add a path of length* $2d - 5$*:*

$$P_v = \{y_1, x_1, y_2, x_2, \ldots, y_{d-3}, x_{d-3}, y_{d-2}\}$$

where each vertex x_i *is of degree 2 and adjacent to the vertices* y_i *and* y_{i+1}*, and each vertex* y_i *is of degree 3 such that* y_1 *is adjacent to* $\{v_1, v_2, x_1\}$*,* y_{d-2} *is adjacent to* $\{v_{d-1}, v_d, x_{d-3}\}$*, and* y_i *is adjacent to* $\{v_{i+1}, x_{i-1}, x_i\}$*, for* $2 \le i \le (d-3)$.

As an example, please refer to the unfolding operation on the vertex v of degree 6 shown in Fig. 2. Note that the unfolding operation does not change the planarity of a graph: the path P_v can be drawn on a small disc on which the vertex v was embedded in $\pi(G)$, and the edges from the vertices $v_1, \ldots, v_d$ to the path P_v can be drawn on the plane without edge crossing.

Suppose we are given a planar graph $G_1 = (V_1, E_1)$, $V_1 = V_{\le 3} \cup V_{>3}$, where $V_{\le 3}$ is the set of vertices whose degree is less than or equal to 3, $V_{>3}$ is the set of vertices whose degree is greater than 3. We apply the unfolding operation on a vertex $v \in V_{>3}$. We get a new planar graph $G_2 = (V_2, E_2)$, where G_2 has one fewer vertex of degree larger than 3, compared with G_1.

We first consider a vertex cover C_2 of the graph G_2.

- Suppose for some i, $1 \le i \le d-3$, the three vertices x_i, y_i, and y_{i+1} are all in C_2. Then we simply remove x_i from C_2. It is obvious that $C_2 - \{x_i\}$ is still a vertex cover of G_2, with one fewer vertex compared with C_2. Call this operation *clean-one*.
- Suppose for some i, $1 \le i \le d-3$, exactly two of the three vertices x_i, y_i, and y_{i+1} are in C_2. If one of these two vertices is x_i, then we can replace the two vertices by y_i and y_{i+1}, resulting in a new vertex cover of the same size. Call this operation *clean-two*.

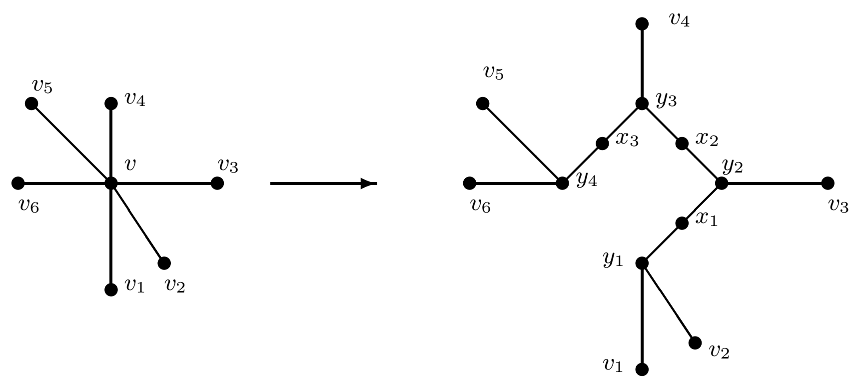

Fig. 2. Unfolding operation on the vertex v (with degree 6).

Note that at least one of the three vertices x_i, y_i, and y_{i+1} must be in the vertex cover C_2 in order to cover the edges $[x_i, y_i]$ and $[x_i, y_{i+1}]$. Therefore, besides the above cases, the only remaining case is that for the three vertices x_i, y_i, and y_{i+1}, only one of them is in C_2. In this case, this vertex in C_2 must be x_i.

In the following discussion, *cleaning* a vertex cover C_2 means that we apply the processing of clean-one and clean-two on C_2. After the cleaning process, we say that the vertex cover C_2 is *clean*. By the above discussion, in a clean vertex cover C_2 of the graph G_2, we have

Claim. Either all $d-3$ vertices x_i, $1 \le i \le d-3$, are in C_2 and none of the $d-2$ vertices y_j, $1 \le j \le d-2$, is in C_2; or all $d-2$ vertices y_j, $1 \le j \le d-2$, are in C_2 and none of the $d-3$ vertices x_i, $1 \le i \le d-3$, is in C_2.

Let C_1 be any vertex cover of the graph G_1 such that C_1 has k_1 vertices. If $v \in C_1$ (so v covers the d edges $[v, v_1]$, …, $[v, v_d]$ in G), then by replacing v in C_1 by the $d-2$ vertices y_1, y_2, …, y_{d-2} in G_2, we obviously get a clean vertex cover C_2 for the graph G_2. The vertex cover C_2 has $k_1 + (d-3)$ vertices. On the other hand, if v is not in C_1 (so the edges $[v, v_1]$, …, $[v, v_d]$ must be covered by the vertices v_1, …, v_d in C_1), then by adding the $d-3$ vertices x_1, x_2, …, x_{d-3} to C_1, we get a clean vertex cover C_2 for the graph G_2 and C_2 contains $k_1 + (d-3)$ vertices. In conclusion, from a vertex cover of k_1 vertices for the graph G_1, we can always construct a (clean) vertex cover of $k_1 + (d-3)$ vertices for the graph G_2.

Conversely, suppose that we are given a clean vertex cover C_2 of the graph G_2, where C_2 has k_2 vertices. If C_2 contains the $d-2$ vertices y_1, y_2, …, y_{d-2}, then replacing the $d-2$ vertices y_1, y_2, …, y_{d-2} in C_2 by a single vertex v gives a vertex cover of $k_2 - (d-3)$ vertices for the graph G_1. On the other hand, if C_2 contains the $d-3$ vertices x_1, x_2, …, x_{d-3}, then removing these $d-3$ vertices from C_2 gives a vertex cover of $k_2 - (d-3)$ vertices for the graph G_1. In conclusion, from a vertex cover of k_2 vertices for the graph G_2, we can always construct a vertex cover of $k_2 - (d-3)$ vertices for the graph G_1.

Now suppose that the set of vertices of degree larger than 3 in the graph G_1 is $V_{>3} = \{u_1, u_2, \ldots, u_r\}$. Denote by $deg(u)$ the degree of the vertex u. Inductively, suppose that the graph G_{i+1} is obtained from the graph G_i by unfolding the vertex u_i, for $1 \le i \le r$. Note that the graph G_r has its degree bounded by 3, and we say that the graph G_r is obtained from the graph G_1 by *unfolding all vertices of degree larger than 3*. Let C_1 be a vertex cover for the graph G_1 with $|C_1| = k_1$. By the above discussion, we can construct from C_1 a vertex cover C_2 of $k_1 + (deg(u_1) - 3)$ vertices for the graph G_2; then from C_2, we can construct a vertex cover C_3 of $k_1 + (deg(u_1) - 3) + (deg(u_2) - 3)$ vertices for the graph G_3, ……, and finally we construct a vertex cover C_r of $k_1 + \sum_{i=1}^{r}(deg(u_i) - 3)$ vertices for the graph G_r.

On the other hand, let C_r be a vertex cover of k_r vertices for the graph G_r. First we clean C_r to get a clean vertex cover C'_r for G_r. Since cleaning does not increase the size of the vertex cover, we have $|C'_r| \le |C_r| = k_r$. Now by the above discussion, we can get a vertex cover C_{r-1} of $|C'_r| - (deg(u_r) - 3) \le k_r - (deg(u_r) - 3)$ vertices for the graph G_{r-1}. Cleaning the vertex cover C_{r-1} gives us a clean vertex cover C'_{r-1} for the graph G_{r-1}, and by the above processing we can get a vertex cover C_{r-2} of $|C'_{r-1}| - (deg(u_{r-1}) - 3) \le k_r - (deg(u_r) - 3) -$

$(deg(u_{r-1}) - 3)$ vertices for the graph G_{r-2}, ..., finally, we will construct a vertex cover of at most $k_r - \sum_{i=1}^{r}(deg(u_i) - 3)$ vertices for the graph G_1.

In particular, the above discussion enables us to derive a relation between the minimum vertex covers for the graphs G_1 and G_r. Let k_1 and k_r be the sizes of minimum vertex covers of the graph G_1 and G_r, respectively. By the above discussion, from a minimum vertex cover for the graph G_1, we can construct a vertex cover of $k_1 + \sum_{i=1}^{r}(deg(u_i) - 3)$ vertices for the graph G_r. Therefore, $k_1 + \sum_{i=1}^{r}(deg(u_i) - 3) \geq k_r$. On the other hand, from a minimum vertex cover of the graph G_r, we can construct a vertex cover of no more than $k_r - \sum_{i=1}^{r}(deg(u_i) - 3)$ vertices for the graph G_1, thus $k_r - \sum_{i=1}^{r}(deg(u_i) - 3) \geq k_1$. Combining these two relations, we get $k_1 + \sum_{i=1}^{r}(deg(u_i) - 3) = k_r$.

Summarizing the above discussion, we get the following:

Claim. Let G_1 be a graph in which the set of vertices of degree larger than 3 is $V_{>3}$. Let G_r be a graph obtained by unfolding all vertices of degree larger than 3 in G_1. Then from a vertex cover C_1 for the graph G_1, we can construct in polynomial time a vertex cover of $|C_1| + \sum_{u \in V_{>3}}(deg(u) - 3)$ vertices for the graph G_r; and from a vertex cover C_r for the graph G_r, we can construct in polynomial time a vertex cover of at most $|C_r| - \sum_{u \in V_{>3}}(deg(u) - 3)$ vertices for the graph G_1. Moreover, the size of a minimum vertex cover of the graph G_r is equal to the size of a minimum vertex cover of the graph G_1 plus $\sum_{u \in V_{>3}}(deg(u) - 3)$.

Using the unfolding operations, we can prove

Lemma 5. *The* P-VC-3 *problem has no parameterized algorithm of time* $2^{o(\sqrt{k})}n^{O(1)}$*, unless the* P-VC *problem has a parameterized algorithm of time* $2^{o(\sqrt{k})}n^{O(1)}$.

Proof. Suppose the P-VC-3 problem has a parameterized algorithm A of time $2^{o(\sqrt{k})}n^{O(1)}$. We have the following algorithm A' shown in Fig 3 for the P-VC problem.

We prove the algorithm A' is correct. By Claim 4, OPT_1 is a vertex cover for the graph G_1 with $|OPT_2| - \sum_{u \in V_{>3}}(deg(u) - 3)$ vertices and OPT_1 is computable in time $n^{O(1)}$. Since OPT_2 is a minimum vertex cover for the graph G_2, by Claim 4 again, a minimum vertex cover for the graph G_1 contains $|OPT_2| - \sum_{u \in V_{>3}}(deg(u) - 3)$ vertices. In conclusion, OPT_1 is a minimum vertex cover for the graph G_1.

We analysis the running time of A' in the following.

For the graph $G_1 = (V_1, E_1)$, $V_1 = V_{\leq 3} \cup V_{>3}$, where $|V_1| = n$ and $|E_1| = m$, we can always assume $|OPT_1| \geq n/2$ by applying the NT-theorem [12]. That is, the parameter $k \geq n/2$. After applying the unfolding operation on each $v \in V_{>3}$, we get the new planar graph $G_2 = (V_2, E_2)$ with degree bounded by 3. The construction of G_2 can be done in polynomial time.

For a planar graph with n vertices and m edges, we have [14]:

$$m \leq 3n - 6. \quad (1)$$

Algorithm A'

Input: A planar graph $G_1 = (V_1, E_1)$, $V_1 = V_{\leq 3} \cup V_{>3}$, and an integer $k > 0$.

Output: Output "Yes", if the size of the minimum vertex cover OPT_1 of G_1 satisfies $|OPT_1| \leq k$. Otherwise, output "No".

1. Let $V_{>3}$ be the set of all vertices of degree larger than 3 in the graph G_1. Construct a planar graph G_2 by unfolding all vertices of degree larger than 3 in G_1.

2. Run the algorithm A on the graph G_2 with the parameter $k_2 = 1, 2, ..., |V_2|$. We get a minimum vertex cover OPT_2 for the graph G_2.

3. Construct a vertex cover OPT_1 for the graph G_1 from OPT_2 such that $|OPT_1| = |OPT_2| - \sum_{u \in V_{>3}} (deg(u) - 3)$.

5. If $|OPT_1| \leq k$, **Return** "Yes"; Otherwise, **Return** "No".

Fig. 3. Parameterized algorithm for PLANAR VERTEX COVER.

By Equation 1, for the graph G_1, the total degree of all its vertices satisfies:

$$\sum_{v \in V_1} deg(v) = 2m \leq 2(3n - 6) < 6n, \tag{2}$$

We have

$$\begin{aligned} |V_2| = |V_{\leq 3}| + \sum_{v \in V_{>3}} ((deg(v) - 3) + (deg(v) - 2)) \\ < |V_{\leq 3}| + 2 \sum_{v \in V_{>3}} deg(v) \\ \leq |V_1| + 2 \sum_{v \in V_1} deg(v) \\ \leq n + 12n = 13n = O(n). \end{aligned}$$

Therefore, the calls to the algorithm A on the graph G_2 takes time $2^{o(\sqrt{|V_2|})}|V_2|^{O(1)} = 2^{o(\sqrt{n})}n^{O(1)} = 2^{o(\sqrt{k})}n^{O(1)}$. All the other steps of the algorithm A' takes polynomial time $n^{O(1)}$. Therefore the algorithm A' has running time $2^{o(\sqrt{k})}n^{O(1)}$.

Therefore, from Lemma 4, Lemma 5 and Theorem 1, we have

Theorem 2. *The* P-VC-3 *problem has no EPTAS of running time* $2^{o(\sqrt{1/\epsilon})}n^{O(1)}$, *where* $\epsilon > 0$ *is the given error bound, unless all SNP problems are solvable in subexponential time.*

Theorem 2 implies the difficulty of improving the EPTAS algorithm for the P-VC-3 problem.

Baker [4] provided an EPTAS algorithm of time $2^{O(1/\epsilon)}p(n)$ for the P-VC problem. By applying that algorithm, we get an EPTAS algorithm of time $2^{O(1/\epsilon)}p(n)$ for the P-VC-3 problem. Since the P-VC-3 problem seems simpler, one might suspect that we could have a better EPTAS algorithm for it than that for the P-VC problem.

In the following we show that if we can improve the EPTAS algorithm for the P-VC-3 problem, then we can improve the EPTAS algorithm for the P-VC problem.

Theorem 3. *If the* P-VC-3 *problem has an EPTAS of running time $f(1/\epsilon)n^{O(1)}$, then the* P-VC *problem has an EPTAS of running time $f(13/\epsilon)n^{O(1)}$, where f is a recursive function and $\epsilon > 0$ is the given error bound.*

Proof. Given an EPTAS algorithm A of running time $f(1/\epsilon)n^{O(1)}$ for the P-VC-3 problem, we provide an EPTAS algorithm B of running time $f(13/\epsilon)n^{O(1)}$ for the P-VC problem. The description of algorithm B is given in Fig. 4.

Algorithm B

Input: A planar graph $G_1 = (V_1, E_1)$, and a constant $\epsilon > 0$.

Output: A vertex cover C_1 for G_1, such that $|C_1| \leq (1+\epsilon) * |OPT_1|$.

1. Let $V_{>3}$ be the set of all vertices of degree larger than 3 in the graph G_1. Unfold all vertices of degree larger than 3 in G_1, let the resulting graph be $G_2 = (V_2, E_2)$, whose degree is bounded by 3.

2. Run the algorithm A with $\epsilon' = \epsilon/13$ on the graph G_2. We get a vertex cover C_2 for the graph G_2.

3. From C_2 construct a vertex cover C_1 of at most $|C_2| - \sum_{u \in V_{>3}}(deg(u) - 3)$ vertices for the graph G_1.

4. **Return** C_1.

Fig. 4. EPTAS algorithm for PLANAR VERTEX COVER.

We claim that the vertex set C_1 is the required vertex cover for the graph G_1.

By Equation 1 and Claim 4, we have

$$\begin{aligned} |OPT_2| &= |OPT_1| + \sum_{u \in V_{>3}} (deg(u) - 3) \\ &\leq |OPT_1| + \sum_{u \in V_1} deg(u) \end{aligned}$$

$$\begin{aligned} &\le |OPT_1| + 6n \\ &\le |OPT_1| + 12|OPT_1| \\ &\le 13|OPT_1|. \end{aligned}$$

Therefore,

$$|OPT_2| \le 13|OPT_1|. \tag{3}$$

By Claim 4, we have

$$|OPT_1| = |OPT_2| - \sum_{u \in V_{>3}} (deg(u) - 3)$$

and

$$|C_1| \le |C_2| - \sum_{u \in V_{>3}} (deg(u) - 3)$$

Therefore, we have

$$|C_2| - |C_1| \ge |OPT_2| - |OPT_1|$$

or equivalently

$$|C_2| - |OPT_2| \ge |C_1| - |OPT_1|$$

From this, we derive immediately

$$\begin{aligned} &|C_1|/|OPT_1| - 1 \\ &= (|C_1| - |OPT_1|)/|OPT_1| \\ &\le (|C_2| - |OPT_2|)/|OPT_1| \\ &\le 13(|C_2| - |OPT_2|)/|OPT_2| \\ &= 13(|C_2|/|OPT_2| - 1) \\ &\le 13 * (\epsilon/13) \\ &= \epsilon. \end{aligned}$$

Here we have used the assumption that $C_2|/|OPT_2| \le 1 + \epsilon' = 1 + \epsilon/13$, and the fact $|OPT_2| \ge 13|OPT_1|$.

The call of the algorithm A on the graph G_2 takes time $f(1/\epsilon')n^{O(1)}$. All the other steps of the algorithm B take polynomial time $n^{O(1)}$. Therefore, the running time of the algorithm B is $f(13/\epsilon)n^{O(1)}$, and the approximation ratio for the algorithm B is $1 + \epsilon$.

5 Summary

In this paper, we have proved lower bound results on the running time of the PTAS algorithms for a class of planar graph problems including PLANAR INDEPENDENT SET, PLANAR VERTEX COVER and PLANAR DOMINATING SET. We pointed out that there is a gap between our lower bound result and the current known upper bound result on the running time of the PTAS algorithms for these planar graph problems. We then studied the PTAS algorithms for PLANAR VERTEX COVER problem. Based on our study of the relationship between PLANAR VERTEX COVER and PLANAR VERTEX COVER on planar graphs of degree bounded by three, we showed that to further improve the upper bound on the running time of the PTAS algorithms for PLANAR VERTEX COVER, we could concentrate on the PLANAR VERTEX COVER on planar graphs of degree bounded by three. Closing the gap and further improving the upper bound on the running time of the PTAS algorithms for these planar graph problems are nice open problems inviting further research.

References

1. Alber J, Bodlaender HL, Fernau H, Kloks T, and Niedermeier R (2002) Fixed parameter algorithms for dominating set and related problems on planar graphs. Algorithmica 33:461-493
2. Alber J, Fernau H, Niedermeier R (2004) Parameterized complexity: exponential speed-up for planar graph problems. J. Algorithms 52:26-56
3. Ausiello G, Crescenzi P, Gambosi G, Kann V, Marchetti-Spaccamela A, and Protasi M (1999) Complexity and Approximation, Combinatorial Optimization Problems and Their Approximability Properties. New York, Springer-Verlag
4. Baker BS (1994) Approximation algorithms for NP-complete problems on planar graphs. Journal of the ACM 41:153-180
5. Bar-Yehuda R and Even S (1982) On approximating a vertex cover for planar graphs. Proceedings of the fourteenth annual ACM symposium on Theory of computing. pp.303-309
6. Cai L and Chen J (1997) On fixed-parameter tractability and approximability of NP optimization problems. Journal Of Computer and System Sciences 54:465-474
7. Cai L, Fellows M, Juedes D, Rosamond F (2006) The complexity of polynomial-time approximation. Theory of Computing Systems, to appear.
8. Cai L and Juedes DW (2003) On the existence of sub-exponential time parameterized algorithms. Journal of Computer and System Sciences 67:789-807
9. Cesati M and Trevisan L (1997) On the efficiency of polynomial time approximation schemes. Information Processing Letters 64:165-171
10. Chen J, Chor B, Fellows M, Huang X, Juedes DW, Kanj I and Xia G (2004) Tight lower bounds for parameterized NP-hard problems. Proc. of the 19th Annual IEEE Conference on Computational Complexity, pp. 150-160
11. Chen J, Huang X, Kanj I and Xia G (2004) Linear FPT reductions and computational lower bounds. Proc. of the 36th ACM Symposium on Theory of Computing, pp. 212-221
12. Chen J, Kanj I, and Jia W (2001) Vertex cover: further observations and further improvements. Journal of Algorithms 41:280-301
13. Chen J, Kanj I, Xia G (2003) A note on parameterized exponential time complexity. Tech. Report, DePaul University
14. Diestel R (2000) Graph theory. New York: Springer
15. Downey RG and Fellows MR (1999) Parameterized complexity. Springer, New York

16. Eppstein D (2000) Diameter and treewidth in minor-closed graph families, Algorithmica 27:275-291
17. Fomin FV and Thilikos DM (2003) Dominating sets in planar graphs: branch-width and exponential speed-up. Proc. of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 168-177
18. Fomin FV and Thilikos DM (2004) A simple and fast approach for solving problems on planar graphs. Lecture Notes in Computer Science 2996:56-67
19. Garey M and Johnson D (1979) Computers and intractability: a guide to the theory of NP-Completeness. W. H. Freeman, New York
20. Grohe M (2003) Local tree-width, excluded minors, and approximation algorithms, Combinatorica 23:613-632
21. Hopcroft JE and Tarjan RE (1974) Efficient planarity testing. Journal of the ACM 21:549-568
22. Impagliazzo R, Paturi R, Zane F (2001) Which problems have strongly exponential complexity? Journal of Computer and System Sciences 63: 512-530
23. Kanj I, Perkovic L (2002) Improved parameterized algorithms for planar dominating set, Lecture Notes in Computer Science 2420:399-410
24. Khanna S, Motwani R (1996) Towards a Syntactic Characterization of PTAS, STOC 1996: 329-337
25. Lipton RJ, Tarjan RE (1980) Applications of a planar separator theorem. SIAM J. Comput. 9:615-627
26. Papadimitriou CH, Yannakakis M (1991) Optimization, approximation, and complexity classes. Journal of Computer and System Sciences 43: 425-440
27. Papadimitriou CH and Yannakakis M (1999) On the complexity of database queries. Journal of Computer and System Sciences 58:407-427