Vertex Cover: Further Observations and Further Improvements*

Jianer Chen†
Department of Computer Science
Texas A&M University
College Station, TX 77843-3112
chen@cs.tamu.edu

Iyad A. Kanj
School of CTI, DePaul University
243 S. Wabash Avenue
Chicago, IL 60604
ikanj@cs.depaul.edu

Weijia Jia
Department of Computer Science
City University of HongKong
HongKong SAR, China
wjia@cs.cityu.edu.hk

Abstract

Recently, there have been increasing interests and progresses in lowering the worst case time complexity for well-known NP-hard problems, in particular for the Vertex Cover problem. In this paper, new properties for the Vertex Cover problem are indicated and several simple and new techniques are introduced, which lead to an improved algorithm of time $O(kn + 1.2852^k)$ for the problem. Our algorithm also induces improvement on previous algorithms for the Independent Set problem on graphs of small degree.

1 Introduction

Many optimization problems from industrial applications are NP-hard. According to the NP-completeness theory [13], these problems cannot be solved in polynomial time unless P = NP. However, this fact does not obviate the need for solving these problems for their practical importance. There has been a number of approaches to attacking the NP-hardness of optimization problems, including approximation algorithms, heuristic algorithms, and average time analysis. Recently, there have also been increasing interest and progress in lowering the exponential running time of algorithms that solve NP-hard optimization problems precisely.

The current paper was motivated by two lines of research on algorithms for NP-hard optimization problems. The first is the recent progress on parameterized algorithms for the Vertex Cover problem (given a graph G and a parameter k, deciding if G has a vertex cover of k vertices), which is central in the study of fixed-parameter tractability theory [9] and has important applications in fields such as computational biochemistry [15]. Buss developed the first fixed-parameter tractable algorithm of running time $O(kn + 2^k k^{2k+2})$ for the problem (see [5]), which was later improved to $O(kn+2^k k^2)$ by Downey and Fellows [10]. More recently, parameterized algorithms for the Vertex

*A preliminary version of this paper was presented at the *25th International Workshop on Graph-Theoretical Concepts in Computer Science* (WG'99), and appeared in *Lecture Notes in Computer Science*, vol. 1665, pp. 313-324.

†This work was supported in part by the National Science Foundation under Grants CCR-9613805 and CCR-0000206. Part of the work was done while this author was visiting City University of HongKong.

Cover problem have further drawn researchers' attention, and continuous improvements on the problem have been developed. Balasubramanian et al. [1] first broke the bound 2 barrier in the base of the exponential term and developed an $O(kn+1.324718^k k^2)$ time algorithm for the problem, which was then slightly improved to an $O(kn+1.31951^k k^2)$ time algorithm by Downey et al. [11]. Niedermeier and Rossmanith further improved the algorithm in [11] by presenting an algorithm of running time $O(kn+1.29175^k k^2)$ [16]. Very recently, this algorithm was slightly improved by Stege and Fellows' algorithm which has running time $O(kn+\max\{1.25542^k k^2, 1.2906^k k\})$ [22]. A general technique was developed by Niedermeier and Rossmanith [17] that can get rid of the polynomial factor from the dominating terms in the complexity of the above algorithms. Thus, for example, the complexity of the algorithms in [16] and [22] can be further improved to $O(kn+1.29175^k)$ and $O(kn+1.2906^k)$, respectively.

The other motivation is from the research on worst case analysis of algorithms for NP-hard optimization problems, in particular for the Independent Set problem. Since the initialization by Tarjan and Trojanowski [23] with an $O(2^{n/3})$ time algorithm for the Independent Set problem, there have been continuous improved algorithms for the problem [3, 14, 19, 20]. The best of these algorithms is due to Robson [19], whose algorithm solves the Independent Set problem in time $O(2^{0.276n})$. Other "efficient" exponential time algorithms for NP-hard optimization problems include Schöning's $O(1.334^n)$ time probabilistic algorithm for the 3-Sat problem [21], Dantsin et al.'s $O(1.481^n)$ time deterministic algorithm for the 3-Sat problem [8], and Beigel and Eppstein's $O(1.3446^n)$ time algorithm for the 3-Coloring problem [4].

In the present paper, we develop a further improved parameterized algorithm for the Vertex Cover problem, starting with two simple but important observations. Our first observation is on the size for the problem kernel [11] and is based on a theorem by Nemhauser and Trotter [18, 2]. We show that in order to decide if a graph has a vertex cover of k vertices, we essentially only need to concentrate on graphs of at most $2k$ vertices. Our second observation is a new, but simple technique to deal with degree-2 vertices, which greatly simplifies the case by case combinatorial analysis. We have also developed a new technique called "iterative branching" which is used to maintain a special structure for a graph so that an efficient branching search procedure is always applicable.

Using the new techniques and observations, we are able to develop improved parameterized algorithms for the Vertex Cover problem. More precisely, we present an $O(kn+1.2852^k)$ time algorithm for the Vertex Cover problem. This improves the previous best parameterized algorithms of running time $O(kn+1.29175^k)$ and $O(kn+1.2906^k)$, by Niedermeier and Rossmanith [16], and Stege and Fellows [22] respectively. Moreover, our algorithm seems conceptually simpler, avoiding many lengthy case by case combinatorial analysis. (Remark. An earlier version of this work [6] claimed a further improvement of the algorithm, which runs in exponential space and $O(kn+1.271^k)$ time, using dynamic programming technique. This further improvement turned out, however, to be incorrect since the technique does not work as indicated.)

We further indicate that our improved parameterized algorithms for the Vertex Cover problem can be employed to develop improved exponential time algorithms for the Independent Set problem on graphs of small degree. For example, our algorithms induce $O(1.174^n)$ and $O(1.201^n)$ time, polynomial-space, algorithms for the Independent Set problem on graphs of

degree bounded by 3 and 4, respectively, while the previous best algorithm for these graph classes is by applying Robson's algorithm [19], which is for the INDEPENDENT SET problem on general graphs and runs in polynomial space and time $O(2^{0.296n}) \approx O(1.227^n)$.

Remark. Beigel [3] recently and independently developed algorithms for the MAXIMUM INDEPENDENT SET problem of running time $O(1.125^n)$ and $O(1.171^n)$ for graphs of degree bounded by 3 and 4, respectively.

2 On problem kernel and degree-2 vertices

Let G be a graph and V' be a subset of vertices in G. In the rest of this paper, we will denote by $G(V')$ the subgraph induced by the vertex set V', i.e., $G(V')$ is a subgraph of G that has vertex set V' and contains all edges in G that have their both ends in V'.

Two standard methods have been employed in the development of efficient parameterized algorithms for the VERTEX COVER problem: *reduction to problem kernel* and *search trees* [9, 11]. Our first observation is on the method of reduction to problem kernel.

Suppose (G, k) is an instance for the VERTEX COVER problem, where G is a graph of n vertices and k is an integer. By *reduction to problem kernel*, we mean we apply a polynomial time preprocessing on the instance (G, k) to construct another instance (G_1, k_1), where G_1 is a smaller graph (the *kernel*) and $k_1 \leq k$, such that G_1 has a vertex cover of k_1 vertices if and only if G has a vertex cover of k vertices. Buss [5] explained a simple algorithm of running time $O(kn)$ that reduces an instance (G, k) for the VERTEX COVER problem to another instance (G_1, k_1), where the graph G_1 has at most k^2 edges (thus at most $2k^2$ vertices) and $k_1 \leq k$. This result has been extensively used in the latter improved parameterized algorithms for the VERTEX COVER problem [1, 11, 16].

We show that the size of the kernel can be further reduced. This is based on a theorem due to Nemhauser and Trotter [18] (see also Bar-Yehuda and Even [2] for a constructive proof).

Proposition 2.1 [NT-Theorem]. *There is an $O(\sqrt{n}m)$ time algorithm that, given a graph G_1 of n vertices and m edges, constructs two disjoint subsets C_0 and V_0 of vertices in G_1 such that*

(1). *Every minimum vertex cover of $G_1(V_0)$ plus C_0 forms a minimum vertex cover for G_1;*

(2). *A minimum vertex cover of $G_1(V_0)$ contains at least $|V_0|/2$ vertices.*

We remark that Proposition 2.1 was proved based on the minimum vertex cover of a bipartite graph of $2n$ vertices and $2m$ edges, which can be constructed in time $O(\sqrt{n}m)$ via a maximum matching of the bipartite graph [2].

We explain how Proposition 2.1 is used to obtain a smaller kernel. Given an instance (G, k) for the VERTEX COVER problem, let (G_1, k_1) be the instance constructed by Buss' algorithm, where G_1 has at most k^2 edges and $k_1 \leq k$. We apply Proposition 2.1 to the graph G_1 to construct the two subsets C_0 and V_0, as described in Proposition 2.1. Now the graph G_1 has a vertex cover of size k_1 if and only if the induced subgraph $G_1(V_0)$ has a vertex cover of size $k_1 - |C_0|$. Since the minimum vertex cover of the graph $G_1(V_0)$ consists of at least $|V_0|/2$ vertices, a necessary condition for the graph $G_1(V_0)$ to have a vertex cover of size $k_1 - |C_0|$ is that the number of vertices $|V_0|$ of the graph $G_1(V_0)$ is bounded by $2k_1 - 2|C_0|$. Let $G_2 = G_1(V_0)$ and $k_2 = k_1 - |C_0|$, then we

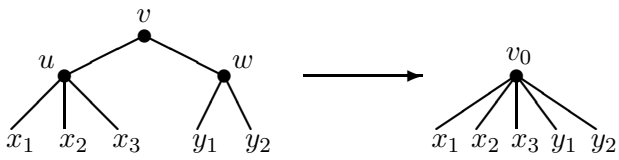


Figure 1: Vertex folding

have constructed an instance (G_2, k_2) for the VERTEX COVER problem, where G_2 has at most $2k_2$ vertices and $k_2 \leq k$, such that the graph G_2 has a vertex cover of k_2 vertices if and only if the graph G_1 has a vertex cover of k_1 vertices, which in consequence is a necessary and sufficient condition for the original graph G to have a vertex cover of k vertices.

Since the graph G_1 has $O(k^2)$ edges, according to Proposition 2.1, the instance (G_2, k_2) can be constructed from (G_1, k_1) in time $O(k^3)$. Summarizing these discussions, we conclude

Theorem 2.2 *There is an algorithm of running time $O(kn + k^3)$ that given an instance (G, k) for the* VERTEX COVER *problem constructs another instance (G_2, k_2), where the graph G_2 contains at most $2k_2$ vertices and $k_2 \leq k$, such that the graph G has a vertex cover of k vertices if and only if the graph G_2 has a vertex cover of k_2 vertices.*

Remark. It seems difficult to further reduce the kernel size. In fact, it is not hard to see that the vertex set $C_0 \cup V_0$ is a vertex cover of the graph G_1 whose size is not larger than twice of that of a minimum vertex cover of G_1. Therefore, further reduction on the kernel size would imply an approximation algorithm of ratio better than 2 for the optimization version of the VERTEX COVER problem, which would resolve a very well-known open problem in approximation algorithms [12].

Our second observation is on the processing of degree-2 vertices during the branching search for a vertex cover of size k. First note that unless a graph G has less than k vertices, which can be easily checked, the graph G has a vertex cover of size exactly k if and only if it has a vertex cover of size bounded by k. Therefore, for each instance (G, k) of the VERTEX COVER problem, we only need to decide if the graph G has a vertex cover of size bounded by k.

There have been several methods proposed in dealing with degree-2 vertices in parameterized algorithms for the VERTEX COVER problem [1, 11, 16]. Most of those methods consider the combinatorial structures case by case and apply different operations according to the combinatorial structures. Here we propose a new method which is simpler and more uniform, and seems more convenient in processing degree-2 vertices.

Suppose v is a degree-2 vertex in the graph G with two neighbors u and w such that u and w are not adjacent to each other. We construct a new graph G' as follows: remove the vertices v, u, and w and introduce a new vertex v_0 that is 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.

Lemma 2.3 *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 the graph G has a vertex cover of size bounded by k if and only if the graph G' has a vertex cover of size bounded by $k - 1$.*

PROOF. Let the two neighbors of the vertex v in the graph G be u and w. Let v_0 be the new vertex in the graph G' that resulted from folding the vertex v.

Suppose S is a vertex cover of size bounded by k in G. Without loss of generality, assume that S is a minimum vertex cover of G. We notice the following facts:

- not all the three vertices v, u, and w are in S;
- if exactly one of v, u, and w is in S, it must be v;
- if exactly two of v, u, and w are in S, we can assume these two are u and w.

Therefore, if v is in S, then $S - \{v\}$ is a vertex cover of the graph G', while if u and w are in S, then $S - \{u, w\} \cup \{v_0\}$ is a vertex cover of the graph G'. In both cases, we have shown that the graph G' has a vertex cover of size bounded by $k - 1$.

For the other direction, it is also easy to see that if the vertex v_0 is in a vertex cover S' of the graph G', then $S' - \{v_0\} \cup \{u, w\}$ is a vertex cover of G, and if v_0 is not in the vertex cover S' then $S' \cup \{v\}$ is a vertex cover of G. □

Therefore, vertex folding reduces the parameter k by 1 directly without any branching. There are another two possible cases when the graph G has a vertex of degree less than 3: (1) the graph G has a degree-1 vertex v. Then we can simply include the neighbor of v in the minimum vertex cover; and (2) the graph G has a degree-2 vertex v whose two neighbors u and w are adjacent to each other. In this case, since a minimum vertex cover S in G must contain at least two vertices in the triangle (v, u, w), we can simply include the vertices u and w in the minimum vertex cover. In both cases, we can do at least as well as vertex folding. Therefore, in the rest discussion in this paper, we will assume, for simplicity and without loss of generality, that whenever the graph G has a vertex of degree 1 or 2, the vertex folding is applicable. Finally, we point out that techniques similar to vertex folding have been implicitly used in previous research, such as in [11].

We will illustrate in the following sections the power of the vertex folding technique.

3 On graphs of degree bounded by 3 and by 4

We first describe a parameterized algorithm for the VERTEX COVER problem on graphs of degree bounded by 3. This algorithm well illustrates the power of the vertex folding technique. For a vertex v, we denote by $N(v)$ the set of neighbors of v.

Let G be a graph in which every vertex is of degree bounded by 3. We also assume that the graph G contains at least one vertex of degree at most 2: if G has no such a vertex, we subdivide an edge in G by two degree-2 vertices. According to Lemma 2.3, the resulting graph G' has a vertex cover of size bounded by $k + 1$ if and only if the graph G has a vertex cover of size bounded by k. We then instead work on the instance $(G', k + 1)$. Consider the algorithm given in Figure 2.[1]

[1]For simplicity, by "including a vertex v in the vertex cover C" we mean "adding v to C, removing v and the edges incident on v from the graph G, and then also removing all vertices of degree 0 in G". Similarly, by "branching at a vertex v_0", we mean we branch into two search paths, one includes the vertex v_0 in the vertex cover C then works recursively on the resulting graph, and the other includes all neighbors of v_0 in C, then works recursively on the resulting graph. These conventions will be used throughout this paper.

VC-Degree-3

1. $C = \emptyset$;
2. **while** $|C| < k$ **and** G is not empty **do**
2.1. pick a degree-2 vertex v in G;
2.2. fold v;
2.3. **if** the new vertex v_0 has degree larger than 2 **then** branch at v_0.

Figure 2: An algorithm for vertex cover on graphs of degree bounded by 3

Since the graph given to each stage (except the first stage) is always a proper subgraph of the original graph G of degree bounded by 3, we ensure that the graph at the beginning of each stage always has a vertex of degree less than 3. The new vertex v_0 is always immediately eliminated in a stage if it has degree larger than 2.

We consider the running time of the algorithm **VC-Degree-3**. Let $C(k)$ be the number of search paths in the search tree of our algorithm searching for a vertex cover of size bounded by k. In step 2.2 of the algorithm **VC-Degree-3**, folding the vertex v reduces the parameter k by 1 (according to Lemma 2.3). When we branch at v_0 in step 2.3, we further reduce the parameter k either by 1 (by including v_0 in the vertex cover), or by $|N(v_0)| \geq 3$ (by including all neighbors of v_0 in the vertex cover). Therefore, we always branch with

$$C(k) \leq C(k-1) \quad \text{or} \quad C(k) \leq C(k-2) + C(k-4)$$

It is easy to verify that $C(k) = \alpha^k$, where $\alpha = 1.272\ldots$ is the root of the polynomial $x^4 - x^2 - 1$. Finally, according to Theorem 2.2, we can assume that the graph G has at most $2k$ vertices (thus $O(k)$ edges). Therefore, each search path takes time $O(k)$. Thus, the running time of the algorithm is bounded by $O(kn + 1.273^k k)$.

Using the above algorithm and the technique in [17], we have the following theorem:

Theorem 3.1 *The* Vertex Cover *problem on graphs of degree bounded by* 3 *can be solved in time* $O(kn + 1.273^k)$.

Remark. Recently, Chen et al. presented an algorithm for the Vertex Cover problem on graphs of degree bounded by 3 of running time $O(kn + 1.2365^k)$ [7].

Now we consider graphs of degree bounded by 4.

Lemma 3.2 *Let v be a vertex of degree 3 in a graph G. Then there is a minimum vertex cover of G that contains either all three neighbors of v or at most one neighbor of v. Moreover, if all neighbors of v have degree at least 3 and there is an edge between two neighbors of v, then we can branch with $C(k) \leq 2C(k-3)$.*

Proof. Let the three neighbors of v be u, w, and z. Suppose that a minimum vertex cover S contains exactly two neighbors u and w of v. Then since S needs to cover the edge (v, z), it must

also contain the vertex v. Now $S - \{v\} \cup \{z\}$ is a minimum vertex cover of G that contains all three neighbors of v.

In particular, if $\{u, w, z\}$ forms a triangle, then we directly derive that there is a minimum vertex cover that contains all u, w, and z.

Now assume that all three neighbors u, w, and z have degree at least 3, and that (u, w) is an edge in G. Moreover, by the previous paragraph, we can assume that $\{u, w, z\}$ does not form a triangle. Observe that since (u, w) is an edge, a minimum vertex cover S of G must contain either u or w. Thus, according to the first part of the lemma, if no minimum vertex cover contains all u, w, and z, then there must be one that does not contain the vertex z (so it must contain all neighbors of z). Thus, we can branch by either including all u, w, and z in the vertex cover, or including all neighbors of z in the vertex cover. Since z has degree at least 3, we conclude that in this case, we can always branch with $C(k) \leq 2C(k-3)$. □

Now we are ready to present our theorem for graphs of degree bounded by 4.

Theorem 3.3 *The* Vertex Cover *problem on graphs of degree bounded by* 4 *can be solved in time* $O(kn + 1.277^k)$.

Proof. We assume that the graph G always has a vertex of degree bounded by 3 — otherwise at the beginning of the process we subdivide an edge of G by two degree-2 vertices. It will be also easy to verify that the property of having a vertex of degree bounded by 3 is preserved throughout our process. Moreover, we also assume that there is at least one degree-4 vertex in G — otherwise we simply apply Theorem 3.1. Again we concentrate on counting the number $C(k)$ of search paths in the search tree of our algorithm to construct a vertex cover of at most k vertices.

If the graph G has a vertex of degree less than 3, then we apply steps 2.2–2.3 of the algorithm **VC-Degree-3** to the vertex. According to our discussion on the algorithm, we can branch with $C(k) \leq C(k-2) + C(k-4)$, which is satisfied by $C(k) = 1.277^k$. Thus, we can further assume that the graph G has no vertex of degree less than 3.

If the graph G has a vertex v of degree 3 such that there is an edge between two neighbors of v, then by Lemma 3.2, we can branch with $C(k) \leq 2C(k-3)$, which is satisfied by $C(k) = 1.277^k$.

Therefore, without loss of generality, we can assume that each vertex in the graph G has degree either 3 or 4, and a vertex v in G has degree 3 with three neighbors u, w, and z, where the vertex z is of degree 4. Moreover, there is no edge between u, w, and z.

Consider the algorithm in Figure 3 (and the figures in Figure 4).

Consider case (A). By Lemma 3.2, we can assume that the minimum vertex cover S either (a1) contains all three vertices u, w, and z, or (a2) contains at most one of them. In case (a1), the parameter k is reduced by 3. In case (a2), both vertices v and t must be in S, reducing the parameter k by 2. Moreover, since all vertices in G have degree either 3 or 4, in case (a2) after including v and t in S, at least two of u, w, and z have degree either 1 or 2. The vertex folding reduces the parameter by 1. The branching at the new vertex r_0 further reduces the parameter k by either 1 or at least 5. Therefore, in case (a2), we either totally reduce the parameter k by 3, or further branch at r_0 so totally the parameter k is reduced either by 4 or by 8. Combining with

VC-Degree-4

case (A). there is a vertex $t \neq v$ that is adjacent to at least two of u, w, and z
branch by (a1) including $\{u, w, z\}$ in the vertex cover;
(a2) including v and t in the vertex cover;
fold a degree-2 vertex r; if the new vertex r_0 has degree > 4, branch at r_0;

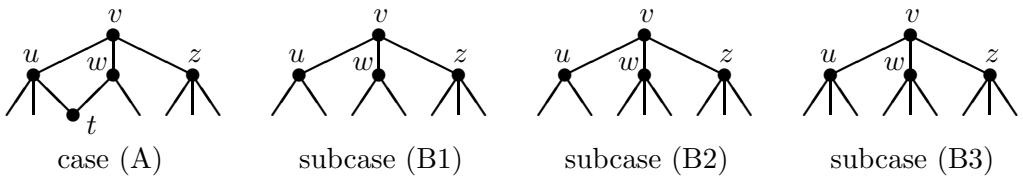
case (B). no vertex, except v, is adjacent to more than one vertex in $\{u, w, z\}$.

subcase (B1). $|N(u) \cup N(w) - \{v\}| = 4$
branch by (b11) including z in the vertex cover and fold v;
(b12) including $N(z)$ in the vertex cover;

subcase (B2). $|N(u) \cup N(w) - \{v\}| = 5$ (suppose $deg(u) = 3$)
branch by (b21) including $\{u, w, z\}$ in the vertex cover;
(b22) including z and $N(u) \cup N(w)$ in the vertex cover;
(b23) including $N(z)$ in the vertex cover;
fold u; if the new vertex u_0 has degree > 4, branch at u_0;

subcase (B3). $|N(u) \cup N(w) - \{v\}| = 6$
branch by (b31) including $\{u, w, z\}$ in the vertex cover;
(b32) including z and $N(u) \cup N(w)$ in the vertex cover;
(b33) including $N(z)$ in the vertex cover;

Figure 3: An algorithm for vertex cover on graphs of degree bounded by 4



case (A) subcase (B1) subcase (B2) subcase (B3)

Figure 4: Vertex cover for a graph of degree bounded by 4

case (a1), we conclude that we branch with

$$C(k) \leq C(k-3) + C(k-3) \quad \text{or} \quad C(k) \leq C(k-3) + C(k-4) + C(k-8) \tag{1}$$

Now consider case (B) in which v is the only vertex adjacent to more than one vertex in u, w, and z. Note that since both u and w have degree either 3 or 4, the only possible values for $|N(u) \cup N(w) - \{v\}|$ are 4, 5, and 6.

In subcase (B1) $|N(u) \cup N(w) - \{v\}| = 4$, we branch at the vertex z. By our assumption, the vertex z is of degree 4. Therefore, in case (b12), we reduce the parameter k by 4. In case (b11) including z in S makes the vertex v have degree 2, so we can further reduce the parameter k by 1 by folding v. No branch at the new vertex v_0 is needed since its degree is 4. Thus, we branch with

$$C(k) \leq C(k-2) + C(k-4) \tag{2}$$

In subcase (B2) $|N(u) \cup N(w) - \{v\}| = 5$, we assume without loss of generality that the degree of u is 3. According to Lemma 3.2, we have either (b21) all u, w, and z are in S; or (b22) z is the only one in S among u, w, z; or (b23) z is not in S. In case (b21), we reduce the parameter k by 3, in case (b22) we reduce the parameter k by $|N(u) \cup N(w) \cup \{z\}| = 7$, and in case (b23), we include $N(z)$ in S and reduce the parameter k by 4. Moreover, in case (b23) including $N(z)$ in S makes the degree of u become 2 so we can fold it, which further reduce the parameter by 1. Now if we branch at u_0, we reduce the parameter k further either by 1 or by at least 5. Thus, we branch with

$$C(k) \leq C(k-3)+C(k-7)+C(k-5) \text{ or } C(k) \leq C(k-3)+C(k-7)+C(k-6)+C(k-10) \tag{3}$$

In subcase (B3) $|N(u) \cup N(w) - \{v\}| = 6$, including all u, w, and z (case (b31)) reduces the parameter k by 3, including only z among u, w, and z (case (b32)) reduces the parameter k by $|N(u) \cup N(w) \cup \{z\}| = 8$, and including $N(z)$ (case (b33)) reduces the parameter k by 4. Thus, we branch with

$$C(k) \leq C(k-3) + C(k-8) + C(k-4) \tag{4}$$

It is easy to verify that $C(k) = 1.277^k$ satisfies all conditions (1)-(4). The theorem now follows. □

4 Iterative branching

In this section we consider graphs of degree bounded by 5. From the previous section, we have seen great advantages for always having a vertex of degree bounded by 3. This condition can be easily satisfied while we are dealing with graphs of degree bounded by 4 since every proper subgraph of a graph of degree bounded by 4 has at least one vertex of degree bounded by 3. However, the condition is no longer guaranteed for graphs of degree bounded by 5. In this section, we introduce a technique, the *iterative branching method*, that imposes degree-3 vertices on graphs of degree bounded by 5, and show how we take advantage of this property.

We say a graph G is a $(5,3)$-*graph* if the maximum vertex degree in G is 5 and G has at least one vertex of degree bounded by 3. Similarly, a $(5,4)$-*graph* has maximum vertex degree 5 and at least one vertex of degree bounded by 4.

Let $C_{5,4}(k)$ be the number of search paths in the search tree of our algorithm to construct a vertex cover of at most k vertices in a $(5,4)$-graph, and let $C_{5,3}(k)$ be the number of search paths in the search tree of our algorithm to construct a vertex cover of at most k vertices in a $(5,3)$-graph. We first list the recurrence relations we need in our analysis.

$$
\begin{aligned}
C_{5,3}(k) &= 2C_{5,3}(k-3) && (5)\\
C_{5,3}(k) &= C_{5,3}(k-2)+C_{5,3}(k-4) && (6)\\
C_{5,3}(k) &= C_{5,3}(k-3)+C_{5,3}(k-4)+C_{5,3}(k-8) && (7)\\
C_{5,3}(k) &= C_{5,4}(k-2) && (8)\\
C_{5,3}(k) &= C_{5,4}(k-2)+C_{5,4}(k-6) && (9)\\
C_{5,3}(k) &= C_{5,4}(k-3)+2C_{5,4}(k-6) && (10)\\
C_{5,3}(k) &= C_{5,4}(k-1) && (11)\\
C_{5,3}(k) &= C_{5,4}(k-2)+C_{5,4}(k-5) && (12)\\
C_{5,3}(k) &= C_{5,4}(k-3)+C_{5,4}(k-5)+C_{5,4}(k-8) && (13)\\
C_{5,3}(k) &= C_{5,4}(k-3)+C_{5,4}(k-6)+C_{5,4}(k-8)+C_{5,4}(k-11) && (14)\\
C_{5,3}(k) &= C_{5,4}(k-2)+C_{5,4}(k-6)+C_{5,4}(k-11) && (15)\\
C_{5,3}(k) &= C_{5,4}(k-3)+C_{5,4}(k-6)+C_{5,4}(k-7)+C_{5,4}(k-12) && (16)\\
C_{5,3}(k) &= C_{5,4}(k-3)+2C_{5,4}(k-7)+2C_{5,4}(k-12) && (17)\\
C_{5,3}(k) &= C_{5,4}(k-3)+C_{5,3}(k-3) && (18)\\
C_{5,4}(k) &= \sum_{i=1}^{q} C_{5,3}(k-5i+4)+C_{5,3}(k-5q) && (19)
\end{aligned}
$$

Our algorithm runs in stages. At the beginning of each stage, we assume that the graph G is a $(5,3)$-graph. We process on a vertex v of degree bounded by 3 in G based on the combinatorial structure of v. The resulting graph G' will be a $(5,4)$-graph. We then apply the iterative branching method on G' to make a $(5,3)$-graph so that the next stage of the algorithm is applicable.

Algorithm. VC-Degree-5

{ The algorithm runs in stages. Assuming G is a $(5,3)$-graph. }

Case 1. The graph G has a vertex v of degree 1.

We include the neighbor of v in the vertex cover, branching with recurrence relation (11).

Case 2. The graph G has a vertex v of degree 2.

If the two neighbors of v are adjacent to each other, we simply include the two neighbors of v in the vertex cover. If the two neighbors of v are not adjacent to each other, then we fold v. In case the new vertex v_0 has degree larger than 4, we also branch at v_0. Note that in case we branch at v_0, we totally reduce the parameter k by either 2 or at least 6. Thus in this case, we branch with recurrence relations (8), (11), or (9).

Excluding cases 1-2, we can assume in the following that all vertices have degree at least 3. Let v be a vertex of degree 3 with neighbors x, y, and z, with z of the largest degree among x, y, and z. Without loss of generality, we can also assume that the degree of z is larger than 3.

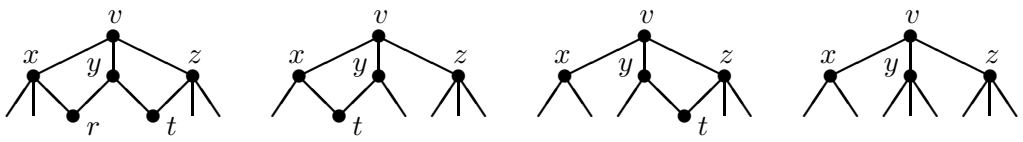


Figure 5: Vertex cover for a graph of degree bounded by 5

Case 3. The degree of z is 5.

We branch at the vertex z. The branch including $N(z)$ in the vertex cover reduces the parameter k by 5. In the branch that includes z in the vertex cover, the vertex v becomes of degree 2, thus we fold v. If the new vertex v_0 has degree larger than 5, we further branch at v_0. Thus in the branch including z, we either reduce the parameter k by 2, or further branch and reduce the parameter k by either 3 or 8. Thus, we branch with the recurrence relations (12) and (13).

Now we further assume that all vertices x, y, and z have degree bounded by 4.

Case 4. There are two edges, say (x, y) and (y, z), among x, y, and z.

Then by Lemma 3.2, there must be a minimum vertex cover that contains y. Thus, we can simply include the vertex y in the vertex cover. We branch with the recurrence relation (11).

Case 5. There is exactly one edge, say (y, z), among x, y, and z.

Then since one of y and z must be in any vertex cover, by Lemma 3.2, we can branch by either including all x, y, and z in the vertex cover (thus reducing the parameter k by 3), or excluding the vertex x (i.e., including $N(x)$ in the vertex cover) thus reducing the parameter k by at least 3. Note that in case of including $N(x)$, the vertex y and z become of degree bounded by 3. Thus we branch with the recurrence relation (18).

Now we can further assume that there is no edge among the vertices x, y, and z.

Case 6. There are two vertices r and t, other than v, that are adjacent to more than one vertex in $\{x, y, z\}$. (see Figure 5).

By Lemma 3.2, there is a minimum vertex cover S that either contains all x, y, and z, or contains at most one of them. In the latter case, all three vertices v, r, and t must be in S. Therefore, we can branch by either including x, y, and z in S, or including v, r, and t in S. Note that in both cases, the resulting graph contains a vertex of degree bounded by 3. This gives the recurrence relation (5).

Case 7. There is exactly one vertex $t \neq v$ adjacent to more than one vertex in $\{x, y, z\}$.

We then branch at z. In case we include z in the vertex cover, the vertex v becomes of degree 2. So we fold v. If the new vertex v_0 has degree larger than 5, we branch at v_0. Therefore, in case we include z, we either reduce the parameter k by 2, or further branch at v_0 that reduces the parameter k totally either by 3 or by at least 8.

On the other hand, if we include $N(z)$, then the resulting graph always has a vertex of degree 2. In fact, we can assume that either t is adjacent to z, or t is adjacent to a degree-3 vertex, say x, among $\{x, y, z\}$ (in case t is adjacent to x and x has degree 4, we can let x be z). In case t is adjacent to x and x has degree 3 (see Figure 5, subcase 7), including $N(z)$ makes the vertex x have degree bounded by 2, while in case t is adjacent to z (see Figure 5, subcase 7'), including $N(z)$ makes either y or x have degree bounded by 2. Therefore, after including $N(z)$, we can fold

a degree-2 vertex, and if the new vertex has degree larger than 5, we further branch at the new vertex. We conclude that in the branch we include $N(z)$, we either reduce the parameter k by 5 (if we do not branch at the new vertex) or further branch at a vertex of degree larger than 5 and reduce the parameter k by either 6 or at least 11. So in this case we branch with the recurrence relations (12), (15), (13), or (14).

Excluding cases 1-7, now we can assume that v is the only vertex that is adjacent to more than one vertex in $\{x, y, z\}$.

Case 8. A vertex in $\{x, y, z\}$, say x, has degree 3.

Then we branch at z. In case we include z, the vertex v becomes of degree 2 (see Figure 5, subcase 8), so we fold v, totally reducing the parameter k by 2. The new vertex v_0 has degree at most 5, so we do not need to further branch at it.

In case we include $N(z)$, the vertex x becomes of degree 2, so we fold x and if the new vertex has degree larger than 5, we further branch at the new vertex. Thus, in this branch, we either reduce the parameter k by 5, or further branch at the new vertex and reduce the parameter k totally by either 6 or at least 11. This case gives the recurrence relations (12) or (15).

Excluding the cases 1-8, we now assume that all vertices x, y, and z have degree exactly 4.

Case 9. A neighbor t of one of x, y, and z has degree 5.

Then we branch at t. In case we include t in the vertex cover, now the situation becomes Case 8, so we apply the algorithm for Case 8. Thus, this branch contributes either $C_{5,4}(k-3)+C_{5,4}(k-6)$ or $C_{5,4}(k-3)+C_{5,4}(k-7)+C_{5,4}(k-12)$.

In case we include $N(t)$ in the vertex cover, the vertex v becomes of degree 2, so we fold v and if the new vertex has degree larger than 5, we further branch at the new vertex. Thus, this branch contributes either $C_{5,4}(k-6)$ or $C_{5,4}(k-7)+C_{5,4}(k-12)$. In summary, in this case we have recurrence relations (10), (16), or (17)

Case 10. All neighbors of x, y, and z have degree at most 4.

We branch at z. In case we include z in the vertex cover, v becomes of degree 2 so we fold it and if the new vertex has degree larger than 5, we further branch at the new vertex. Note that in this case, the neighbors of z become of degree bounded by 3, so the resulting graph is a $(5, 3)$-graph.

In case we include $N(z)$, the degree of x and y becomes bounded by 3 and the resulting graph is a $(5, 3)$-graph. Thus, the recurrence relations are (6) and (7).

end of Algorithm VC-Degree-5.

It is easy to verify that in all cases in the algorithm **VC-Degree-5**, the resulting graph G has a vertex of degree bounded by 4. In case the graph G has all its vertices of degree bounded by 4, we simply apply Theorem 3.3. Otherwise, G is a $(5, 4)$-graph. We now apply the iterative-branch method to G. See Figure 6 for the algorithm.

On each degree-4 and degree-5 vertex pair (v, u), the algorithm **Iterative-Branch** branches at the degree-5 vertex u. In case the degree-5 vertex u is included in the vertex cover, the vertex v becomes of degree 3 so the algorithm *stops* the process; while in case the neighbors $N(u)$ of the vertex u are included in the vertex cover, the algorithm continues the process by working on another pair of degree-4 and degree-5 vertices. Therefore, at the end of the algorithm, we should end up either with a graph of degree bounded by 4, to which we can apply Theorem 3.3 directly, or with a

Iterative-Branch

while the graph G is a $(5,4)$-graph and has no vertices of degree ≤ 3 **do**
pick a vertex v of degree 4 that is adjacent to a vertex u of degree 5;
branch with (1) including u in the vertex cover, and **STOP**;
(2) including $N(u)$ in the vertex cover.

Figure 6: The iterative branch algorithm

$(5,3)$-graph so that the next stage of the algorithm **VC-Degree-5** is applicable. In particular, the algorithm **Iterative-Branch** gives recurrence relation (19) for $C_{5,3}$ and $C_{5,4}$, where the integer q in (19) is the number of times the **while** loop body in algorithm **Iterative-Branch** is executed.

To solve the recurrence relations (5)-(19), we introduce a special polynomial. Let h be a positive integer and a_i, $1 \leq i \leq h-1$, be nonnegative real numbers with at least one a_i strictly larger than 0. We call the following polynomial a *branching polynomial*.

$$p_b(x) = x^h - \sum_{i=0}^{h-1} a_i x^i$$

Lemma 4.1 *A branching polynomial $p_b(x)$ has exactly one positive root α_0. Moreover, for any $\alpha > 0$, $p_b(\alpha) \leq 0$ if and only if $\alpha \leq \alpha_0$;*

Proof. Let $p_b(x) = x^h - \sum_{i=0}^{h-1} a_i x^i$ be the branching polynomial. Since the leading coefficient of $p_b(x)$ is positive, there is a constant β such that $p_b(x) > 0$ for all $x \geq \beta$. Moreover, since there is at least one a_i strictly larger than 0, there is an ϵ, $0 < \epsilon < \beta$, such that $p_b(\epsilon) < 0$ and $p_b(x)$ has no root in the interval $(0, \epsilon)$. The conditions $p_b(\epsilon) < 0$ and $p_b(\beta) > 0$ give immediately that $p_b(x)$ has at least one positive root.

We prove by induction on the degree h of $p_b(x)$ that $p_b(x)$ has only one positive root. This is obviously true for $h = 1$. Now consider the case $h \geq 2$. Suppose by contradiction that $p_b(x)$ has more than one positive root. From the inequalities $p_b(\epsilon) < 0$ and $p_b(\beta) > 0$, and the fact that the polynomial $p_b(x)$ has more than one root in the interval (ϵ, β), we can derive easily that the derivative $p_b'(x)$ of $p_b(x)$ has at least two positive roots. This contradicts the inductive hypothesis since $p_b'(x)/h$ is also a branching polynomial.

The second part of the lemma follows directly from the inequalities $p_b(\epsilon) < 0$ and $p_b(\beta) > 0$, and the fact that α_0 is the unique positive root of $p_b(x)$. □

We have the following lemma for the recurrence relations (5)-(18).

Lemma 4.2 *Each of the recurrence relations (5)-(18) has solution $C_{5,3}(k) = \alpha^k$, where $\alpha \leq 1.285$.*

Proof. Recurrence relations (5)-(7) involve in only the function $C_{5,3}$ so it is easy using standard technique to check that the lemma holds for them. The correctness of the lemma on (8) is implied by that on (11). The correctness of the lemma on (9) is implied by that on (12).

Therefore, we only need to prove the lemma for recurrence relations (10)-(18).

Consider recurrence relation (10): $C_{5,3}(k) = C_{5,4}(k-3) + 2C_{5,4}(k-6)$. Applying formula (19) to the functions $C_{5,4}(k-3)$ and $C_{5,4}(k-6)$ in (10), we obtain

$$\begin{aligned} C_{5,3}(k) &= \sum_{i_1=1}^{q_1} C_{5,3}(k-5i_1+1) + C_{5,3}(k-5q_1-3) + \sum_{i_2=1}^{q_2} C_{5,3}(k-5i_2-2) + C_{5,3}(k-5q_2-6) \\ &\quad + \sum_{i_3=1}^{q_3} C_{5,3}(k-5i_3-2) + C_{5,3}(k-5q_3-6) \end{aligned} \tag{20}$$

Note that the two occurrences $C_{5,4}(k-6)$ in (10) may end up with different numbers q_2 and q_3 of times the **while** loop body in algorithm **Iterative-Branch** is executed.

It is well-known that the recurrence relation (20) has a solution $C_{5,3}(k) = \alpha^k$, where α is the root of the characteristic polynomial $p_b^{(10)}(x)$ of the recurrence relation (20) [1, 16], where:

$$p_b^{(10)}(x) = x^k - \sum_{i_1=1}^{q_1} x^{k-5i_1+1} - x^{k-5q_1-3} - \sum_{i_2=1}^{q_2} x^{k-5i_2-2} - x^{k-5q_2-6} - \sum_{i_3=1}^{q_3} x^{k-5i_3-2} - x^{k-5q_3-6} \tag{21}$$

Rearranging the characteristic polynomial $p_b^{(10)}(x)$, and using the standard formula for the summation of geometric series, we obtain

$$p_b^{(10)}(x) = \frac{x^{k-5q_1-3}}{x^5-1}\left(x^{5q_1+1}(x^7-x^3-x^2-2) - (x^{5q_1-5q_2-3} + x^{5q_1-5q_3-3} + 1)(x^5-x^4-1)\right)$$

It can be easily verified that $\alpha_{10} = 1.2844\ldots$ is a root of the polynomial $(x^7-x^3-x^2-2)$. Note the polynomial x^5-x^4-1 has a root $1.32\ldots$ so by Lemma 4.1, we have $\alpha_{10}^5 - \alpha_{10}^4 - 1 < 0$. Thus, $p_b^{(10)}(\alpha_{10}) > 0$. Since the polynomial $p_b^{(10)}(x)$ is a branching polynomial, by Lemma 4.1 again, we conclude that the root of $p_b^{(10)}(x)$ is not larger than $\alpha_{10} = 1.2844\ldots < 1.285$.

The proofs of the lemma for the recurrence relations (11)-(18) are very similar. In general, we apply formula (19) to each of the recurrence relations to obtain a recurrence relation containing only the function $C_{5,3}$, then we construct the corresponding characteristic polynomial, which is a branching polynomial. We list these characteristic polynomials as follows.

$$\begin{aligned}
p_b^{(11)}(x) &= \frac{x^{k-5q-1}}{x^5-1}\left(x^{5q+1}(x^5-x^3-1) - (x^5-x^4-1)\right) \\
p_b^{(12)}(x) &= \frac{x^{k-5q_1-2}}{x^5-1}\left(x^{5q_1+1}(x^6-x^3-x-1) - (x^{5q_1-5q_2-3}+1)(x^5-x^4-1)\right) \\
p_b^{(13)}(x) &= \frac{x^{k-5q_1-3}}{x^5-1}\Big(x^{5q_1-1}(x^9-x^5-x^4-x^3-1) \\
&\qquad - (x^{5q_1-5q_2-2} + x^{5q_1-5q_3-5} + 1)(x^5-x^4-1)\Big) \\
p_b^{(14)}(x) &= \frac{x^{k-5q_1-3}}{x^5-1}\Big(x^{5q_1-4}(x^{12}-x^8-x^7-x^5-x^3-1) \\
&\qquad - (x^{5q_1-5q_2-3} + x^{5q_1-5q_3-5} + x^{5q_1-5q_4-8} + 1)(x^5-x^4-1)\Big) \\
p_b^{(15)}(x) &= \frac{x^{k-5q_1-2}}{x^5-1}\Big(x^{5q_1-5}(x^{12}-x^9-x^7-x^5-1)
\end{aligned}$$

$$
\begin{aligned}
& - (x^{5q_1-5q_2-4} + x^{5q_1-5q_3-9} + 1)(x^5 - x^4 - 1)\Big) \\
p_b^{(16)}(x) &= \frac{x^{k-5q_1-3}}{x^5-1}\Big(x^{5q_1-5}(x^{13} - x^9 - x^8 - x^6 - x^5 - 1) \\
& - (x^{5q_1-5q_2-3} + x^{5q_1-5q_3-4} + x^{5q_1-5q_4-9} + 1)(x^5 - x^4 - 1)\Big) \\
p_b^{(17)}(x) &= \frac{x^{k-5q_1-3}}{x^5-1}\Big(x^{5q_1-5}(x^{13} - x^9 - x^8 - 2x^5 - 2) \\
& - (x^{5q_1-5q_2-4} + x^{5q_1-5q_3-4} + x^{5q_1-5q_4-9} + x^{5q_1-5q_5-9} + 1)(x^5 - x^4 - 1)\Big) \\
p_b^{(18)}(x) &= \frac{x^{k-5q-3}}{x^5-1}\Big(x^{5q}(x^8 - x^5 - x^4 - x^3 + 1) - (x^5 - x^4 - 1)\Big)
\end{aligned}
$$

where $p_b^{(i)}$ is the characteristic polynomial for recurrence relation (i), $11 \leq i \leq 18$. The constants q_j in each polynomial are the constant q in formula (19) when we replace each copy of a $C_{5,4}$ in the recurrence relation using the formula. Each of the characteristic polynomials has the form

$$
p_b^{(i)}(x) = \frac{x^{k-5q+d_1}}{x^5-1}\Big(x^{5q+d_2}p_i(x) - (x^{c_1} + \cdots + x^{c_r} + 1)(x^5 - x^4 - 1)\Big)
$$

We then verify that the polynomial $p_i(x)$ has a root α_i such that $1 < \alpha_i \leq 1.285$. This implies that $p_b^{(i)}(1.285) \geq 0$. By Lemma 4.1, we conclude that the positive root of the branching polynomial $p_b^{(i)}(x)$ is not larger than 1.285. □

Theorem 4.3 *The* Vertex Cover *problem on graphs of degree bounded by* 5 *can be solved in time* $O(kn + 1.285^k)$.

Proof. Let (G, k) be an instance of the Vertex Cover problem, where G is a graph of degree bounded by 5. According to Theorem 2.2, we can assume that the graph G has at most $2k$ vertices, thus $O(k)$ edges. If G is a 5-regular graph, then we let G' be a graph obtained by subdividing an edge of G by two degree-2 vertices. Note that G' is a $(5,3)$-graph so algorithm **VC-Degree-5** is applicable. Moreover, by Lemma 2.3, the graph G has a vertex cover of at most k vertices if and only if the graph G' has a vertex cover of at most $k+1$ vertices. Thus, instead we can work on the instance $(G', k+1)$ with G' a $(5,3)$-graph. If the graph G' has degree bounded by 4, then Theorem 3.3 guarantees the correctness of the theorem. Otherwise, by Lemma 4.2, we have $C_{5,3}(k) \leq 1.285^k$, so the Vertex Cover problem on the instance $(G', k+1)$ can be solved in time $O(C_{5,3}(k+1)(k+1)) = O(1.285^{k+1}(k+1)) = O(1.285^k k)$. Again, using the technique in [17], the theorem follows. □

5 Putting all together

Now we are ready to describe the entire algorithm.

Given a graph G, we first apply Theorem 2.2 to reduce it to its kernel. Then we branch at each vertex v of degree at least 6 with recurrence relation

$$
C(k) \leq C(k-1) + C(k-6)
$$

Note that the function $C(k) = 1.2852^k$ satisfies this recurrence relation. The running time of this stage is $O(kn + 1.2852^k k^2)$, which can be reduced to $O(kn + 1.2852^k)$ using the technique in [17].

After eliminating all vertices of degree larger than 5, we get a graph of degree bounded by 5. Now combining this with Theorem 4.3, which gives an $O(kn + 1.285^k)$ time algorithm for the Vertex Cover problem on graphs of degree bounded by 5, we conclude with an $O(kn + 1.2852^k)$ time algorithm for the Vertex Cover problem on general graphs.

Theorem 5.1 *The* Vertex Cover *problem can be solved in time* $O(kn + 1.2852^k)$.

Theorem 5.1 is a clear improvement over the previous best algorithms of time $O(kn + 1.29175^k)$ and $O(kn + 1.2906^k)$ for the Vertex Cover problem [16, 22, 17].

6 Improving algorithms for Independent Set

The Maximum Independent Set problem — given a graph, find a maximum independent set — has been playing a major role in the study of optimization problems. Initialized by Tarjan and Trojanowski's algorithm of running time $O(2^{n/3})$ [23], efficient exponential time algorithms for the Maximum Independent Set problem have been investigated for more than two decades. Jian [14] refined Tarjan and Trojanowski's algorithm and presented an algorithm of running time $O(2^{0.304n})$, and Shindo and Tomita [20] developed more recently a simpler algorithm of running time $O(2^{n/2.863})$. The best algorithm for the Maximum Independent Set problem is due to Robson [19] whose algorithm for solving the Maximum Independent Set problem runs in time $O(2^{0.276n})$. This bound has stood as the best for about two decades.

We show a different approach to solving the Maximum Independent Set problem via the Vertex Cover problem. We show that using the algorithms we have developed for the Vertex Cover problem, we can obtain improved algorithms for the Maximum Independent Set problem on graphs of small degree. We first prove the following lemma.

Lemma 6.1 *Let G be a connected graph of n vertices and degree bounded by d. Then a minimum vertex cover of G contains at most $(n(d-1)+1)/d$ vertices.*

Proof. We construct an independent set I for G by repeatedly applying the process of including a vertex v of the smallest degree in I and removing v and its neighbors from the graph. Except for the first vertex in I, which may have degree d, all other vertices picked by the process have degree at most $d-1$. Therefore, the total number of vertices in I is at least $\lceil (n-1)/d \rceil$. Moreover, the set I is obviously an independent set in G. Now the complement of I is a vertex cover of G and has at most $n - \lceil (n-1)/d \rceil \leq n - (n-1)/d = (n(d-1)+1)/d$ vertices. □

Theorem 6.2 *For graphs of degree bounded by 3, there is an algorithm of running time* $O(1.174^n)$ *that solves the* Maximum Independent Set *problem.*

Proof. Let G be a graph of degree bounded by 3. The graph G may not necessarily be connected.

Let the connected components of G be $C_1, \ldots, C_k$, of size $n_1, \ldots, n_k$, respectively. It is clear that a maximum independent set of G is the union of maximum independent sets of the components $C_1, \cdots, C_k$.

For each component C_i of G, instead of finding a maximum independent set for C_i, we try to construct a vertex cover of k vertices, for $k = 1, 2, \ldots$. At the first k for which we are able to construct a vertex cover of k vertices for C_i, we know this vertex cover is a minimum vertex cover. Thus, the complement of this vertex cover is a maximum independent set for C_i. By Lemma 6.1, we must have $k \leq (2n_i + 1)/3$. Thus, by Theorem 3.1, a maximum independent set for the component C_i can be constructed in time $O(n_i(n_i + 1.273^{(2n_i+1)/3}))$, which is bounded by $O(1.174^{n_i})$. In conclusion, a maximum independent set in the graph G can be constructed in time $O(1.174^{n_1}+\cdots+1.174^{n_k})$. Now it is fairly straightforward to verify that $O(1.174^{n_1}+\cdots+1.174^{n_k}) = O(1.174^n)$. □

Similarly, we have

Theorem 6.3 *There is an algorithm of running time $O(1.201^n)$ that solves the* Maximum Independent Set *problem on graphs of degree bounded by* 4.

Remark. The current best algorithm using polynomial space for the Maximum Independent Set problem is due to Robson [19], which has time complexity $O(2^{0.296n}) \approx O(1.227^n)$. Robson has also presented an exponential space algorithm of running time $O(2^{0.276n}) \approx O(1.211^n)$ for the problem [19]. Theorem 6.2 and Theorem 6.3 give polynomial space algorithms of improved time complexity, for graphs of degree bounded by 3 and 4. Based on the results and techniques in the current paper, the time complexity for the Maximum Independent Set problem for graphs of degree bounded by 3 has been further improved to $O(1.152^n)$ [7].

Acknowledgments. The authors are grateful to Richard Beigel, Leizhen Cai, David Eppstein, Mike Fellows, Rolf Niedermeier, Mike Robson, and Peter Rossmanith for their comments and discussions on earlier drafts of this paper, and would like to thank the anonymous referees for their comments, suggestions, and corrections.

References

[1] R. Balasubramanian, M. R. Fellows, and V. Raman, An improved fixed parameter algorithm for vertex cover, *Information Processing Letters* **65**, (1998), pp. 163-168.

[2] 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.

[3] R. Beigel, Finding maximum independent sets in sparse and general graphs, *Proc. 10th Annual ACM-SIAM Symposium on Discrete Algorithms* (SODA'99), (1999), pp. 856-857.

[4] R. Beigel and D. Eppstein, 3-coloring in time $O(1.3446^n)$: a no-MIS algorithm, *Proc. 36th IEEE Symposium on Foundations of Computer Science* (FOCS'95), (1995), pp. 444-452.

[5] J. F. Buss and J. Goldsmith, Nondeterminism within P, *SIAM Journal on Computing* **22**, (1993), pp. 560-572.

[6] J. Chen, I. A. Kanj, and W. Jia, Vertex cover: further observations and further improvements, *Proc. 25th International Workshop on Graph-Theoretical Concepts in Computer Science* (WG'99), *Lecture Notes in Computer Science* **1665**, (1999), pp. 313-324.

[7] J. Chen, L. Liu, and W. Jia, Improvement on Vertex Cover for low-degree graphs, *Networks* **35**, (2000), pp. 253-259.

[8] E. Dantsin, A. Goerdt, E. Hirsch, and U. Schöning, Deterministic algorithms for k-SAT based on covering codes and local search, *Lecture Notes in Computer Science* **1853**, (2000), pp. 236-247.

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

[10] R. G. Downey and M. R. Fellows, Parameterized computational feasibility, in *Feasible Mathematics II*, P. Clote and J. Remmel, eds., Boston, Birkhauser (1995), pp. 219-244.

[11] R. G. Downey, M. R. Fellows, and U. Stege, Parameterized complexity: A framework for systematically confronting computational intractability, in *Contemporary Trends in Discrete Mathematics: From DIMACS and DIMATIA to the Future*, F. Roberts, J. Kratochvil, and J. Nesetril, eds., *AMS-DIMACS Proceedings Series* **49**, AMS, (1999), pp. 49-99.

[12] D. Hochbaum, Approximating covering and packing problems: set cover, vertex cover, independent set, and related problems, in *Approximation Algorithms for NP-hard Problems*, D. Hochbaum, ed., PWS Publishing Company, Boston, (1997), pp. 94-143.

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

[14] T. Jian, An $O(2^{0.304n})$ algorithm for solving maximum independent set problem, *IEEE Trans. Comput.* **35**, (1986), pp. 847-851.

[15] I. A. Kanj, Vertex Cover: exact and approximate algorithms and applications, *Ph.D. Dissertation*, Dept. Computer Science, Texas A&M University, College Station, Texas, (2001). (Available from `http://www.cs.tamu.edu/people/iakanj/research.html`).

[16] R. Niedermeier and P. Rossmanith, Upper bounds for vertex cover further improved, *Lecture Notes in Computer Science* **1563** (STACS'99), (1999), pp. 561-570.

[17] R. Niedermeier and P. Rossmanith, A general method to speed up fixed-parameter-tractable algorithms, *Information Processing Letters* **73**, (2000), pp. 125-129.

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

[19] J. M. Robson, Algorithms for maximum independent sets, *Journal of Algorithms* **7**, (1986), pp. 425-440.

[20] M. Shindo and E. Tomita, A simple algorithm for finding a maximum clique and its worst-case time complexity, *Sys. and Comp. in Japan* **21**, (1990), pp. 1-13.

[21] U. Schöning, A probabilistic approach for k-SAT and constraint satisfaction problems, *Proc. 40th IEEE Symposium on Foundations of Computer Science* (FOCS'99), (1999), pp. 410-414.

[22] U. Stege and M. Fellows, An improved fixed-parameter-tractable algorithm for vertex cover, *Technical Report* **318**, Department of Computer Science, ETH Zurich, April 1999.

[23] R. E. Tarjan and A. E. Trojanowski, Finding a maximum independent set, *SIAM Journal on Computing* **6**, (1977), pp. 537-546.