Improvement on Vertex Cover for Low-Degree Graphs

Jianer Chen
College of Information Engineering, Central-South University of Technology, ChangSha, Hunan, People's Republic of China; Department of Computer Science, Texas A&M University, College Station, Texas 77843-3112

Lihua Liu, Weijia Jia
College of Information Engineering, Central-South University of Technology, ChangSha, Hunan, People's Republic of China; Department of Computer Science, City University of Hong Kong, Hong Kong

We present an improved algorithm for the Vertex Cover problem on graphs of degree bounded by 3 (3DVC). We show that the 3DVC problem can be solved in time $O(1.2192^k k)$, where k is the number of vertices in a minimum vertex cover of the graph. Our algorithm also improves previous algorithms on the Independent Set problem on graphs with degree bounded by 3. © 2000 John Wiley & Sons, Inc.

Keywords: vertex cover; independent set; algorithm; parameterized computation

1. INTRODUCTION

The Vertex Cover problem (given a graph G and an integer k, decide if G has a vertex cover of k vertices) is one of the six "basic" NP-complete problems according to Garey and Johnson [7]. It is fixed-parameter tractable in terms of Downey and Fellows [5], that is, it can be solved in time $O(f(k)n^c)$, where f is a function of the parameter k but independent of the input length n and c is a constant independent of the parameter k. Continuous efforts have been made in improving fixed-parameter tractable algorithms for the vertex cover problem, starting from Buss' algorithm of running time $O(kn + 2^k k^{2k+2})$ [2]. Downey and Fellows [4] developed an $O(kn + 2^k k^2)$ time algorithm. More recently, Balasubramanian et al. [1] developed an $O(kn + 1.3247^k k^2)$ time algorithm. Later Downey et al. [6] improved the result slightly to time complexity $O(kn + 1.3195^k k^2)$, and Niedermeier and Rossmanith [9] developed an algorithm of time $O(kn + 1.2918^k k^2)$. More recently, Chen et al. [3] developed an $O(kn + 1.271^k k^2)$ time algorithm. For the restricted version of the vertex cover problem on graphs of degree bounded by 3 (denoted as 3DVC), the best algorithm known so far is by Chen et al. [3], with running time $O(1.2497^k k)$.

A closely related problem is the time complexity for the Independent Set problem. The first algorithm for the independent set problem was developed by Tarjan and Trojanowski, with time complexity $O(2^{n/3})$ [12]. The best algorithm for the independent set problem so far is due to Robson [11], with time complexity $O(2^{0.276n}) = O(1.21^n)$. For the independent set on graphs of degree bounded by 3, the best-known algorithm is due to Chen et al. [3], with running time $O(1.161^n)$.

Most algorithms for the vertex cover problem are based on the method of bounded search tree [4, 6]. On a vertex v of degree d, we branch by either including v in the vertex cover or excluding v from the vertex cover. In the latter case, all neighbors of v must be in the vertex cover. Thus, branching at a vertex of degree d either adds 1 vertex or adds d vertices in the vertex cover. Let $C(k)$ be the number of search paths in the search tree of our algorithm to find a vertex cover of k vertices; then, the above branching gives us the recurrence relation $C(k) = C(k-1) + C(k-d)$. It is not hard to see that the value of $C(k)$ is decreased when the value d increases. Therefore, the bottleneck of improving algorithms for the vertex cover problem is on graphs of low degree (see [1, 3, 9]). This situation is also true for algorithms for independent set problem (see [11, 12]).

Motivated by the above observation, we concentrate on the 3DVC problem in this paper and present an $O(1.2192^k k)$ time algorithm for the problem. The algorithm can be easily applied to the independent set problem on graphs of degree bounded by 3, resulting in an $O(1.1504^n)$ time algorithm for the problem. Both of our algorithms improve the previous best results [3].

Received June 1999; accepted November 1999
Correspondence to: J. Chen

2. PRELIMINARIES

Let G be a graph. A set C of vertices in G is a *vertex cover* of G if every edge in G has at least one end in the set C. A set I of vertices in G is an *independent set* if no two vertices in I are adjacent. It is easy to see that for any graph $G = (V, E)$ a set C of vertices is a vertex cover of G if and only if $V - C$ is an independent set in G. The 3DVC problem is for a given pair (G, k), where G is a graph of degree bounded by 3 and k is an integer, to decide if G has a vertex cover of k vertices. The Maximum Independent Set problem is for a given graph G to construct an independent set of maximum number of vertices. Both problems are well-known NP-hard problems [7].

Let (G, k) be an instance for the 3DVC problem. Without loss of generality, we assume that the graph G is connected. Let n and m be the number of vertices and number of edges in G, respectively. It is not hard to see that if G has a vertex cover of size k then $n = O(k)$ and $m = O(k)$. As indicated by Chen et al. using a theorem by Nemhauser and Trotter [10] (see [3], Proposition 2.1 and Theorem 2.2), there is an algorithm of running time $O(m\sqrt{n}) = O(\sqrt{k^3})$ that constructs another instance (G_1, k_1) for the 3DVC problem, where G_1 has at most $2k_1$ vertices and $k_1 \leq k$, such that G has a vertex cover of k vertices if and only if G_1 has a vertex cover of k_1 vertices. We summarize this discussion in the following proposition:

Proposition 2.1. *There is an algorithm of running time $O(\sqrt{k^3})$ that given an instance (G, k) for* 3DVC *constructs another instance (G_1, k_1) for* 3DVC, *where G_1 has at most $2k_1$ vertices and $k_1 \leq k$, such that G has a vertex cover of size k if and only if G_1 has a vertex cover of size k_1.*

By Proposition 2.1, we only need to concentrate on instances (G, k) of 3DVC where the graph G has at most $2k$ vertices when we develop fixed-parameter tractable algorithms for 3DVC.

Suppose that 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.

Proposition 2.2 [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 $k - 1$.*

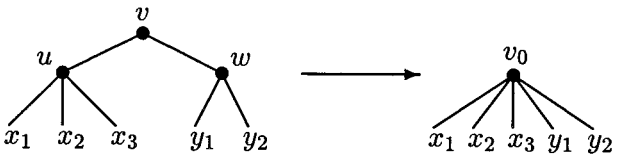

FIG. 1. Vertex folding.

Folding a degree-2 vertex reduces the parameter k by 1 "for free." The other case for a degree-2 vertex v is that the two neighbors u and w of v are adjacent. In this case, since every vertex cover of the graph G contains at least two of the three vertices v, u, and w, we can simply include the two vertices u and w in the objective vertex cover and reduce the parameter k by 2. Thus, in this case, we can do even better than folding. To simplify our discussion, therefore, we will assume without loss of generality in the rest of this paper that when the graph G has a degree-2 vertex the vertex folding technique is always applicable.

3. THE ALGORITHM

Our algorithm is performed in stages. At the beginning of each stage, we assume the following condition on our graph G:

Degree Condition. The graph G has all its vertices of degree bounded by 3, and either has two nonadjacent degree-2 vertices or has a degree-1 vertex.

Note that if a connected graph G has more than three vertices with three of them of degree bounded by 2 then the graph G must satisfy the Degree Condition.

We make some remarks: We assume that our original graph is connected. However, as our algorithm proceeds, at each stage, we may have the graph with more than one connected component. Suppose that (G, k) is an instance given to a stage of our algorithm, where G satisfies the Degree Condition. If a connected component C of G has fewer than, say, 20 vertices, we can simply find the minimum vertex cover of C by the brute-force method. This will reduce the parameter from k to k' with $k' < k$. However, the resulting graph $G' = G - C$ may no longer satisfy the Degree Condition. To reinstate the Degree Condition, we pick a connected component C' of G' with at least four vertices and subdivide an edge of C' by two degree-2 vertices. Let the resulting graph be G''. Since G' is a proper subgraph of the original graph, G' must have at least one vertex of degree less than 3. Therefore, the graph G'' satisfies the Degree Condition. According to Proposition 2.2, the graph G' has a vertex cover of size k' if and only if the graph G'' has a vertex cover of size $k' + 1$, which, in consequence, is the necessary and sufficient condition for the graph G to have a vertex cover of k vertices. Thus, the instance (G, k) has been replaced by the instance $(G'', k' + 1)$, where the pa-

rameter is not increased: $k' + 1 \leq k$, and the graph G'' has one fewer small connected components than has G. By repeating this process, we can construct an instance, without increasing the parameter, in which the graph has no connected component of less than 20 vertices.

This observation suggests to us to make the following assumption without loss of generality:

Size Assumption. At any stage in our algorithm, each connected component of our graph has at least 20 vertices.

Our algorithm is given in Figure 2. Here, by "including a vertex v in the vertex cover," we mean that we add the vertex v to the objective vertex cover, remove the vertex v and all edges incident on v, and then also remove all vertices of degree 0 in the graph.

A subroutine **Branch** is called by our algorithm. When the subroutine **Branch**(v_0) is called on a degree-3 vertex v_0, the given graph G is a 3-regular graph; while when the subroutine **Branch**(v_0) is called on a degree-4 vertex v_0, all other vertices in the graph G have degree bounded by 3. The subroutine **Branch**(v_0) works based on the combinatorial structures of the vertex v_0, reinstates the Degree Condition, then recursively calls the algorithm **New-3dVC** on the resulting graph.

Description of the Subroutine Branch(r)

Suppose that the subroutine **Branch**(r) is called on a vertex r, where the vertex r has degree either 3 or 4. Let the set of neighbors of r be $N(r)$. Moreover, let $N^2(r)$ be the set of vertices, excluding $\{r\} \cup N(r)$, that are adjacent to a vertex in $N(r)$. We further split the set $N^2(r)$ into two sets $N_c^2(r)$ and $N_o^2(r)$, where $N_c^2(r)$ (the "closed set") is the subset of vertices in $N^2(r)$ that are adjacent only to vertices in $N(r)$, and $N_o^2(r)$ (the "open set") is the subset of vertices in $N^2(r)$ that are also adjacent to vertices not in $N(r)$. Figure 3(A) illustrates the structures of these sets.

According to the degree constraint, the set $N_c^2(r)$ contains at most three vertices.

We pick a proper vertex v and consider branching by either including v or including all neighbors of v in the objective vertex cover. First note that if we branch at the vertex v of degree at least 3 then the search path that includes v in the vertex cover has all neighbors of v (there are at least 3) become of degree 1 or 2, thus reinstating the Degree Condition. Therefore, we only need to concentrate on how to select a vertex v of degree at least 3 so that the search path including the neighbors of v also reinstates the Degree Condition.

If the subset $N_o^2(r)$ has more than two vertices, then the search path including the neighbors of r in the vertex cover makes all vertices in the subset $N_o^2(r)$ become of degree 1 or 2, thus reinstating the Degree Condition. If $N_o^2(r)$ has exactly two vertices x and y but x and y are not adjacent, then the search path including $N(r)$ in the vertex cover will make x and y into two nonadjacent vertices of degree 1 or 2, thus reinstating the Degree Condition. Finally, if any vertex x in $N_o^2(r)$ is adjacent to two vertices in $N(r)$, or the vertex x has degree 2, then the search path including the neighbors of r in the vertex cover makes the vertex x of degree 1, thus also reinstating the Degree Condition.

Therefore, we only need to consider the case in which $N_o^2(r)$ has one vertex or two adjacent vertices, and each vertex in $N_o^2(r)$ has degree 3 and is adjacent to exactly one vertex in $N(r)$.

```
Algorithm New-3dVC
{ Assume that the graph G satisfies the Degree Condition }
1. while G has a vertex of degree 3 do
       if G has a vertex v of degree 1
       then include the neighbor of v in the objective vertex cover
       else if G has a vertex v of degree 2
           then fold v;
                if the new vertex v0 has degree 4 and no neighbor of v0 is of degree 1
                then call subroutine Branch(v0)
           else { G is a 3-regular graph }
                pick any degree-3 vertex v0, call subroutine Branch(v0)
2. { at this point, all vertices of G have degree bounded by 2 }
   construct the minimum vertex cover directly.
```

FIG. 2. The main algorithm for 3DVC.

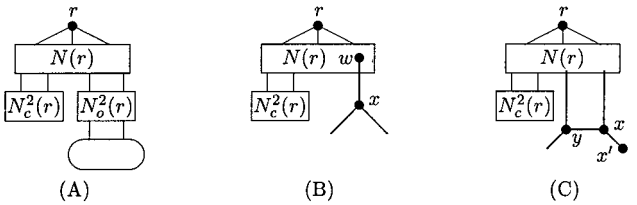

FIG. 3. The combinatorial structures of $N(r)$, $N_c^2(r)$, and $N_o^2(r)$.

CASE 1. The set $N_o^2(r)$ has exactly one vertex x, which is of degree 3. Suppose that its unique neighbor in $N(r)$ is w. [See Fig. 3(B) for illustration.]

Then, we branch at the vertex x (instead of at r).

In the search path including the vertex x, the subgraph H_1 induced by the vertex set $\{r\} \cup N(r) \cup N_c^2(r)$ becomes an isolated piece of at most eight vertices. Thus, we can construct the minimum vertex cover of H_1 and add them directly to the objective vertex cover. Since each vertex in H_1 had degree at least 2 before the vertex x is removed, it is easy to see that a minimum vertex cover of H_1 has at least two vertices. Therefore, in this search path, we can add at least three vertices to the objective vertex cover and reduce the parameter k by at least 3. Let the resulting instance be (G_1, k_1), where $k_1 \leq k-3$. Now, the graph G_1 may not satisfy the Degree Condition. For this, we subdivide an edge of G_1 by two degree-2 vertices. Let the resulting graph be G_2, which satisfies the Degree Condition (note that the graph G_1 has at least one vertex of degree less than 3). By Proposition 2.2, we can recursively work on the instance $(G_2, k_1 + 1)$, where $k_1 + 1 \leq k - 2$. In conclusion, the search path including the vertex x reduces the parameter k by at least 2 and always reinstates the Degree Condition.

In the search path including the neighbors of x, the subgraph H_2 induced by the vertex set $\{r\} \cup N(r) \cup N_c^2(r) - \{w\}$ becomes isolated. Thus, we can add at least one more vertex to the objective vertex cover by constructing the minimum vertex cover of the graph H_2. This reduces the parameter k totally by at least 4. Now, we subdivide an edge by two degree-2 vertices in the resulting graph, which increases the parameter by 1 but reinstates the Degree Condition. Therefore, this search path reduces the parameter k by at least 3.

We should also point out that in the above process the vertex r is always eliminated. Therefore, the resulting graph has no vertex of degree larger than 3.

Summarizing the above discussion, we conclude that when the subroutine **Branch**(r) is called, in Case 1, we can branch with the recurrence relation

$$C(k) = C(k-2) + C(k-3), \tag{1}$$

where $C(k)$ is the number of search paths in the search tree for our algorithm to construct a vertex cover of k vertices.

In the discussion of Case 1, we actually have introduced an interesting technique: Suppose that at some moment we have added d vertices to the objective vertex cover but the Degree Condition no longer holds. Then, we can reinstate the Degree Condition by "paying back" a vertex from the objective vertex cover. This is done as follows: Suppose that the graph already has at least one vertex of degree less than 3. Then, by subdividing an edge in the graph by two degree-2 vertices, we obtain a graph of at least three vertices of degree bounded by 2, thus reinstating the Degree Condition. The cost of this reinstation is that the parameter k is increased by 1.

CASE 2. The set $N_o^2(r)$ has exactly two vertices x and y, both of degree 3 such that (x, y) is an edge. [See Fig. 3(C) for illustration.]

We further split the case into two subcases: The two vertices x and y are either adjacent to the same vertex in $N(r)$ or adjacent to different vertices in $N(r)$. We start by considering the first subcase:

Suppose that the vertices x and y are adjacent to the same vertex w in $N(r)$ (see Fig. 4, Case 2.1). Then, we branch at w (instead of at r). In the search path including w in the objective vertex cover, the subgraph H_3 induced by $\{r\} \cup N_c^2(r) \cup N(r) - \{w\}$ becomes isolated, so we can directly include a minimum vertex cover of H_3 into the objective vertex cover. Since each vertex in $N(r) - \{w\}$ has degree at least 2 in H_3, a minimum vertex cover of H_3 has size at least 2. Therefore, we totally include at least three vertices in the objective vertex cover. If the Degree Condition does not hold now, then, as we explained for Case 1, we can increase the parameter by 1 to reinstate the Degree Condition. In conclusion, in the search path including w in the objective vertex cover, we can reduce the parameter by at least 2 and reinstate the Degree Condition. Now, consider the search path including $N(w)$ in the objective vertex cover [note that $|N(w)| = 3$]. After including $N(w)$, the subgraph H_4 induced by $(N(r) - \{w\}) \cup N_c^2(r)$ becomes isolated so its minimum vertex cover (of size at least 1) can be included into the objective vertex cover directly. Reinstating the Degree Condition may cost us 1 vertex. In summary, in the search path including $N(w)$, we can reduce the parameter by at least 3 and reinstate the Degree Condition.

Summarizing the discussion, in case x and y are adjacent to the same vertex in $N(r)$, we can branch with the recurrence relation (1) and reinstate the Degree Condition. Moreover, the vertex r, which is the only vertex of degree possibly larger than 3, is always eliminated.

Now, consider the second subcase in which the vertices x and y are adjacent to different vertices in $N(r)$. Let w, y, and x' be the three neighbors of x, where $w \in N(r)$ and $y, x' \notin N(r)$. Then, we branch at the vertex x (instead of at r).

Consider the search path that includes x in the objective vertex cover. The vertex w becomes of degree 1 or 2. If w is of degree 1, we simply add r to the objective vertex cover. This makes all vertices in $N(r) - \{w\}$ (there are at least two) become of degree 1 or 2. Since the vertex x' also has degree less than 3, the Degree Condition

has been reinstated and the parameter k is reduced by 2. Also note that the vertex r is eliminated. Now, assume that w is of degree 2 after including x in the objective vertex cover. If w is adjacent to another vertex u in $N(r)$ (see Fig. 4, Case 2.2), then we can directly add vertices r and u in the objective vertex cover. If w is adjacent to a vertex t in $N_c^2(r)$ (see Fig. 4, Case 2.3), then we can fold the vertex w, which reduces the parameter k by 1 and reduces the degree of r by 1 (note the vertex t has degree at least 2). Note that this makes the vertex r have degree at most 3. We, then, if necessary, reinstate the Degree Condition by increasing the parameter by 1. Therefore, in the search path, we include the vertex x in the objective vertex cover; we can always eliminate the vertex of degree 4, reduce the parameter k by at least 1, and reinstate the Degree Condition.

Now, consider the search path that includes the three neighbors of x in the objective vertex cover. Note that in this case the subgraph H_5 induced by the vertex set $\{r\} \cup N(r) \cup N_c^2(r) - \{w\}$ becomes an isolated piece (see Fig. 4, Case 2.4). Since each vertex in $N(r)$ has degree at least 2 before we include $N(x)$, it can be easily checked that if the degree of r was 3 then the minimum vertex cover of H_5 has at least one vertex, while if the degree of r was 4, then the minimum vertex cover of H_5 has at least two vertices. Therefore, after including $N(x)$ in the objective vertex cover, we can add at least $deg(r) - 2$ more vertices to the objective vertex cover, where $deg(r)$ is the degree of r before inclusion of $N(x)$. Now, if necessary, we increase the parameter by 1 to reinstate the Degree Condition. In conclusion, in this branch, if $deg(r) = 4$, then we reduce the parameter by 4, while if $deg(r) = 3$, then we reduce the parameter by 3. Moreover, the Degree Condition is reinstated.

In summary, in this subcase, we branch with the recurrence relation

$$C(k) = \begin{cases} C(k-1) + C(k-3) & \text{if } deg(r) = 3 \\ C(k-1) + C(k-4) & \text{if } deg(r) = 4. \end{cases} \quad (2)$$

As we pointed out before, besides Cases 1 and 2, the subroutine **Branch**(r) can simply branch at the vertex r and reinstate the Degree Condition. In this case, the branching recurrence relation is the same as in (2). Summarizing all these discussions, we conclude [note that the recurrence relation $C(k) = C(k-2) + C(k-3)$ is covered by the recurrence relation $C(k) = C(k-1) + C(k-4)$ and $C(k) = C(k-1) + C(k-3)$] the following:

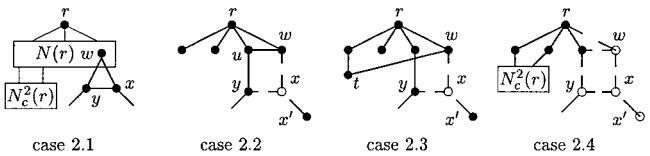

FIG. 4. The structure when $N_o^2(r)$ contains two adjacent vertices x and y.

Lemma 3.1. *The subroutine* **Branch**(r) *can always branch with the recurrence relation* (2) *and reinstate the Degree Condition.*

Theorem 3.2. *The algorithm* **New-3dVC** *runs in time* $O(1.2365^k k)$ *and solves* 3D*VC.*

Proof. According to Proposition 2.1, after an $O(\sqrt{k^3})$ time preprocessing, we can assume that the graph G has at most $2k$ vertices and $O(k)$ edges. The preprocessing time $O(\sqrt{k^3})$ can be simply ignored sine $\sqrt{k} \le 1.2365^k$ for all $k \ge 1$. To initialize the algorithm, we need that the given graph G satisfies the Degree Condition, which, in general, may not be the case. We may simply make the Degree Condition true by subdividing an edge of G by four degree-2 vertices. According to Proposition 2.2, the resulting graph G' has a vertex cover of size bounded by $k+2$ if and only if the original graph G has a vertex cover of size bounded by k. We can thus start our algorithm with the instance $(G', k+2)$.

The algorithm **New-3dVC** deals with degree-1 and degree-2 vertices "for free" until it calls the subroutine **Branch**(v_0).

Since the graph given at the beginning of the algorithm satisfies the Degree Condition, the parameter is reduced by at least 1 before the subroutine **Branch**(v_0) is called. Moreover, if the graph G at beginning has a degree-1 vertex v, removing the neighbor of v introduces at least another vertex of degree less than 3. Therefore, the parameter will be further reduced by at least 1 before the subroutine **Branch**(v_0) is called; while if the graph G at the beginning has two nonadjacent degree-2 vertices, then folding one of them should still leave the other one of degree 2. In particular, if the subroutine **Branch**(v_0) is called on a vertex of degree 3, then the parameter must have been reduced by at least 2 before this call.

Therefore, according to Lemma 3.1, if the subroutine **Branch**(v_0) is called on a degree-4 vertex v_0, then we branch with the recurrence relation

$$C(k) \le C(k-2) + C(k-5), \quad (3)$$

while if the subroutine **Branch**(v_0) is called on a degree-3 vertex v_0, then we branch with the recurrence relation

$$C(k) \le C(k-3) + C(k-5). \quad (4)$$

Also, according to Lemma 3.1, since the subroutine **Branch**(v_0) always reinstates the Degree Condition, the next stage of the algorithm **New-3dVC** is applicable to the new graph.

It can be easily verified that the recurrence relations (3) and (4) have a solution $C(k) = 1.2365^k$. Since the algorithm takes time $O(k)$ on each search path, we conclude that the algorithm **New-3dVC** has running time $O(1.2365^k k)$. ∎

4. FURTHER IMPROVEMENT AND FINAL REMARKS

Further improvement of our algorithm can be achieved using the method suggested in [3].

Let G be a graph of degree bounded by 3. Pick a constant α such that $0 < \alpha < 1$. We first count how many subsets V' of vertices of G are there such that the induced subgraph $G(V')$ is connected and has a vertex cover of size bounded by αk. According to Proposition 2.1, we can assume that the subset size $|V'|$ is bounded by $2\alpha k$. Each induced connected subgraph $G(V')$ of at most $2\alpha k$ vertices can be represented by a binary tree as follows: Order the edges of G in an arbitrary order. Pick any vertex v in $G(V')$ and any edge e incident to v [e does not have to be in $G(V')$]. Label the root of the binary tree by v (and e). Now, the edges incident to v correspond to the children of v in the binary tree. If a vertex w adjacent to v is in $G(V')$, then the corresponding child is labeled by w, and if the vertex w adjacent to v is not in $G(V')$, then the corresponding child is a leaf. It is easy to see that given a tree and a vertex v and an edge e, a connected induced subgraph is uniquely determined. Therefore, the number of connected subgraphs induced by at most $2\alpha k$ vertices in G is bounded by $2k \cdot 3 \cdot \beta_2 = 6k\beta_2$, where β_2 is the number of different binary trees of $2\alpha k$ vertices. It is known (see [8], p. 584) that

$$\beta_2 = \begin{pmatrix} 2 \cdot 2\alpha k \\ 2\alpha k \end{pmatrix} \frac{1}{(2-1)2\alpha k + 1} \leq \begin{pmatrix} 4\alpha k \\ 2\alpha k \end{pmatrix} \frac{1}{2\alpha k}.$$

Using the Stirling approximation [8], we have $\beta_2 \leq c_2 16^{\alpha k}/\sqrt{k^3}$, where c_2 is a small constant. Therefore, the number of connected subgraphs induced by at most $2\alpha k$ vertices in G is bounded by $O(16^{\alpha k}/\sqrt{k})$. Now, for each such an induced subgraph, we apply our algorithm to check whether it has a vertex cover of size bounded by αk. If so, we record it. For each such an induced subgraph, checking this takes time $O(1.2365^{\alpha k}k)$. Therefore, the total time for checking this for all connected subgraphs induced by a vertex subset of at most $2\alpha k$ vertices is bounded by

$$O((16^{\alpha k}/\sqrt{k}) \cdot 1.2365^{\alpha k} \cdot k) = O(16^{\alpha k} 1.2365^{\alpha k} \cdot k).$$

Now, suppose that we have decided the value of α. Then, we modify our algorithm as follows: Given an instance (G, k) for the 3DVC problem, we first enumerate all connected subgraphs of G with at most $2\alpha k$ vertices and check if each of them has a vertex cover of size bounded by αk. By the above discussion, this can be done in time $O(16^{\alpha k} 1.2365^{\alpha k} k)$. Then, we apply the algorithm in Theorem 3.2 to the graph G *but to find at most* $k - \alpha k$ *vertices in the vertex cover.* Once we have found at least $k - \alpha k$ vertices in the vertex cover, we apply Proposition 2.1 (to make sure that the resulting graph has at most $2\alpha k$ vertices) and check directly whether the remaining graph has a vertex cover of size bounded by αk. In this case, the running time of our algorithm is given by

$$O(1.2365^{k-\alpha k}k + 16^{\alpha k}1.2365^{\alpha k}k).$$

Choosing α so that $1.2365^{k-\alpha k} = 16^{\alpha k}1.2365^{\alpha k}$, we obtain $\alpha \approx 0.066398$. With this choice of α, our algorithm's running time is reduced to $O(1.2365^{k-0.066398k}k) = O(1.2192^k k)$.

Theorem 4.1. *The* 3DVC *problem can be solved in time* $O(1.2192^k k)$

Theorem 4.1 also hints an improvement on the independent set problem, as described below.

It is easy to see that every graph of degree bounded by 3 and n vertices has an independent set of at least $n/3$ vertices. In consequence, a minimum vertex cover of the graph contains at most $2n/3$ vertices.

Theorem 4.2. *For graphs of degree bounded by* 3, *there is an algorithm of running time* $O(1.1413^n)$ *that solves the maximum independent set problem.*

Proof. Given a graph of degree bounded by 3, instead of finding a maximum independent set, 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, we know this vertex cover is a minimum vertex cover. Thus, the complement of this vertex cover is a maximum independent set. According to the above observation, we have $k \leq 2n/3$. Therefore, the running time of this algorithm is bounded by $O(1.2192^{2n/3}) = O(1.1413^n)$. ∎

This gives an improvement on the previous best algorithm of running time $O(1.161^n)$ (see [3]).

REFERENCES

[1] R. Balasubramanian, M. R. Fellows, and V. Raman, An improved fixed parameter algorithm for vertex cover, Info Process Lett 65 (1998), 163–168.

[2] J. F. Buss and J. Goldsmith, Nondeterminism within P, SIAM J Comput 22 (1993), 560–572.

[3] J. Chen, I. A. Kanj, and W. Jia, Vertex cover: Further observation and further improvements, Lecture Notes in Computer Science 1665 (WG'99), Springer, Berlin, 1999, pp. 313–324.

[4] R. G. Downey and M. R. Fellows, "Parameterized computational feasibility," Feasible mathematics II, P. Clote and J. Remmel, (Editors), Birkhauser, Boston, 1995, pp. 219–244.

[5] R. G. Downey and M. R. Fellows, Parameterized complexity, Springer-Verlag, New York, 1999.

[6] R. G. Downey, M. R. Fellows, and U. Stege, "Parameterized complexity: A framework for systematically confronting computational intractability," Contemporary trends in discrete mathematics: From DIMACS and DIMATIA to the future, F. Roberts, J. Kratochvil, and J.

Nesetril, (Editors), AMS-DIMACS Proceedings Series 49, AMS, 1999, pp. 49–99.
[7] M. Garey and D. Johnson, Computers and intractability: A guide to the theory of NP-completeness, Freeman, San Francisco, 1979.
[8] D. E. Knuth, The art of computer programming, Addison-Wesley, Reading, MA, 1968, Vol. 1.
[9] R. Niedermeier and P. Rossmanith, "Upper bounds for vertex cover further improved," Lecture Notes in Computer Science 1563 (STACS'99), Springer, Berlin, 1999, pp. 561–570.
[10] G. L. Nemhauser and L. E. Trotter, Vertex packing: Structural properties and algorithms, Math Program 8 (1975), 232–248.
[11] J. M. Robson, Algorithms for maximum independent sets, J Alg 7 (1986), 425–440.
[12] R. E. Tarjan and A. E. Trojanowski, Finding a maximum independent set, SIAM J Comput 6 (1977), 537–546.