Improved Upper Bounds for Partial Vertex Cover*

Joachim Kneis, Alexander Langer, Peter Rossmanith

Dept. of Computer Science, RWTH Aachen University, Germany

Abstract. The Partial Vertex Cover problem is to decide whether a graph contains at most k nodes covering at least t edges. We present deterministic and randomized algorithms with run times of $O^*(1.396^t)$ and $O^*(1.2993^t)$, respectively. For graphs of maximum degree three, we show how to solve this problem in $O^*(1.26^t)$ steps. Finally, we give an $O^*(3^t)$ algorithm for Exact Partial Vertex Cover, which asks for at most k nodes covering exactly t edges.

1 Introduction

The widely known problems Vertex Cover and Dominating Set are among the most important graph-theoretical optimization problems: Find a small set of nodes that cover all edges, or dominate the whole graph, respectively. These NP-complete problems are well studied with respect to approximability [13, 16], exact algorithms [8, 18], and parameterized complexity [6, 7]. Recently, *partial* variants of these and similar problems came into a broader research focus [2–5, 9–12, 14, 15]: Instead of covering all edges or dominating all nodes, it is sufficient to cover t edges or dominate t nodes, where t is an additional parameter. Being generalizations of Vertex Cover and Dominating Set, these problems are NP-complete.

In this paper, we study the complexity of Partial Vertex Cover defined as:

Input: A graph $G = (V, E)$, positive integers k, t
Question: Is there a $C \subseteq V$, $|C| \leq k$, such that C covers at least t edges?

The best known constant approximation factor with respect to k is 2 and there are several algorithms that achieve an approximation factor of $2 - o(1)$ [2, 4, 9, 11, 12]. Since this coincides with the best result for Vertex Cover (see, e.g., [16]), a significant improvement seems to be unlikely. With respect to t, it is easy to see that a simple greedy algorithm already has an approximation ratio of at least 2.

We can expect that Partial Vertex Cover is a harder problem than Vertex Cover: Many algorithms exploit the fact that if each edge $\{u, v\}$ must be covered, one of u or v must be part of the solution. This simple observation

* Supported by the DFG under grant RO 927/7-1

already gives us a 2-approximation and an $O^*(2^k)$ algorithm.[1] In the partial case, however, one does not know if an edge is being covered at all. Thus, the 2-approximation factor for Partial Vertex Cover is harder to prove, and we cannot expect an $O^*(f(k))$ algorithm for Partial Vertex Cover, as this implies FPT = W[1] [10].

The more interesting question is whether there is an fpt algorithm for the case that t rather than only k is small, i.e., an algorithm with run time bounded by $f(t)poly(n)$. Bläser answered this question positively even for Partial Set Cover [3], which is a generalization of Partial Vertex Cover. His randomized algorithm is based on color coding, achieving a run time of $O^*(5.437^t)$, and can be derandomized into a deterministic algorithm. The base in the exponential function is rather huge, though. A faster and much simpler randomized algorithm developed recently [14] achieves a run time of $O^*(2.09^t)$. While this is a significant improvement, derandomizing it would result in a time complexity that is not exponential in t.

In this paper, we present a deterministic algorithm with run time bounded by $O^*(1.396^t)$, which even beats the best known randomized methods. As the latter are based on the *many witnesses* paradigm, they cannot directly be efficiently derandomized. We overcome this obstacle by a new method that scans the possible witnesses in a special order. This way, either a good witness is found early on or the time spent on false witnesses is small. For graphs of maximum degree three, we devise a special algorithm with run time $O^*(1.26^t)$.

Moreover, we present a randomized algorithm for Partial Vertex Cover with a run time bounded by $O^*(1.2993^t)$ improving all previous results. While the algorithm is very simple — it basically selects either a node of maximum degree or two of its neighbors — the analysis is rather involved.

We also consider the variant of Partial Vertex Cover, where *exactly* t edges must be covered and introduce the new technique of *random orientations*. A randomized algorithm based on this technique solves this variant with a run time bounded by $O^*(3^t)$.

Due to space constraints, some of the proofs are omitted in the paper.

2 Preliminaries

Let $G = (V, E)$ be a graph and $U = \{v_1, \ldots, v_u\} \subseteq V$. For $v \in V$, the set of neighbors of v is denoted by $N(v)$, and $N[v] := N(v) \cup \{v\}$. By $\deg(U)$ we denote the degree sequence $(d_1, \ldots, d_u) = (\deg(v_{i_1}), \deg(v_{i_2}), \ldots, \deg(v_{i_u}))$, where $(i_1, \ldots, i_u)$ is a permutation of $(1, \ldots, u)$, such that $d_1 \geq d_2 \geq \cdots \geq d_u$. By $E(U)$ we denote the set of edges that are incident to some $v \in U$, and $||U|| := |E(U)|$. We call $C \subseteq V$ a (t, k)-vertex cover for G iff $|C| \leq k$ and $||C|| \geq t$. We define the relation $\succ$ on all instances of Partial Vertex Cover as $(G, k, t) \succ (G', k', t')$ iff $t > t'$ or $t = t' \wedge |G| > |G'|$.

A *branching vector* $(x_1, x_2, \cdots, x_l)$ is a short notation for a recursive function $T(n)$ of the form $T(n) = T(n - x_1) + T(n - x_2) + \cdots + T(n - x_l)$ for $n > 1$ and

[1] The O^* notation suppresses polynomial factors.

$T(n) = 1$ for $n \leq 1$. The corresponding *branching number* c can be used to bound $T(n)$ by $O(c^n)$ and can easily be computed using characteristic polynomials. For more information about branching vectors and the corresponding branching numbers, see [17].

Let $s = (s_1, \ldots, s_l)$ and $t = (t_1, \ldots, t_l)$ be two branching vectors. We say s dominates t (denoted by $s \trianglerighteq t$ or $t \trianglelefteq s$), iff $s_i \geq t_i$ for $1 \leq i \leq l$. If $s \trianglerighteq t$, then the branching number for s is smaller than the branching number for t.

Let $u, v \in V$ be adjacent nodes of degree at least two. If $N[v] \subseteq N[u]$, we say u *dominates* v. We call G *reduced*, if there are no such nodes in G. For Partial Vertex Cover, we can assume G is reduced without loss of generality, otherwise the operation depicted in Figure 1 can be applied.

Fig. 1. Domination in graphs can be resolved by small modifications.

The following lemmata can easily be deduced from a simple node exchange argument.

Lemma 1. *Let $G = (V, E)$ a graph and v a node of maximum degree d. If $v \notin C$ for* any *(t, k)-vertex cover $C \subseteq V$, then for each (t, k)-vertex cover C holds $i := |C \cap N(v)| > d - d_i + 1$, where $\deg(C \cap N(v)) = (d_1, \ldots, d_i)$.*

Lemma 2. *Let G be a graph, v be a node of maximum degree d and $N(v) = \{v_1, \ldots, v_d\}$, such that $\deg(v_1) \geq \cdots \geq \deg(v_d)$. If there is some i, such that for all (t, k)-vertex cover C we have $C \cap \{v, v_1, \ldots, v_i\} = \emptyset$, but $\deg(v_i) \leq i$, then G does not contain any (t, k)-vertex cover at all.*

Lemma 3. *Let G be a graph, v be a node of maximum degree d and C a (t, k)-vertex cover for G. Let $N(v) = \{v_1, \ldots, v_d\}$ with $\deg(v_1) \geq \cdots \geq \deg(v_d)$. If there is no (t, k)-vertex cover containing any node from $v, v_1, \ldots, v_{d-2}$, then there is a (t, k)-vertex cover C' for G containing both v_{d-1} and v_d and we have $\deg(v_{d-1}) + \deg(v_d) > d$.*

3 A Fast Algorithm on Graphs of Maximum Degree Three

In this section, we present a new deterministic algorithm for Partial Vertex Cover on graphs of maximum degree three with a run time bounded by $O^*(1.26^t)$. The algorithm always branches on a node of degree three and some of its neighbors, thereby avoiding any node that has three neighbors of degree

three if possible. Since we are only forced to select such a node in a three-regular graph, this can be avoided in any but the first step.

The branching itself depends on the degree of the neighbors and the edges between them, leading to a large case distinction. In order to increase the readability, we do not present the algorithm explicitly, but describe its behavior in the upcoming lemmata. Lemma 4 shows the possible branching operations if there is some triangle containing a node of degree three. Lemma 5 establishes the branching operations needed in triangle-free graphs of maximum degree three.

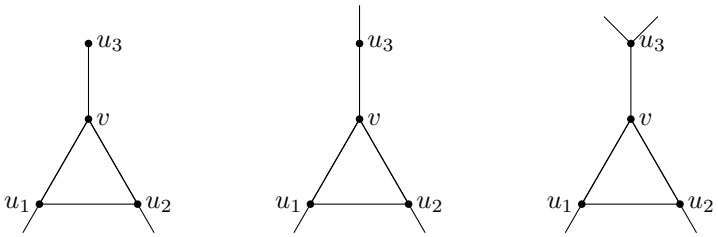

Fig. 2. Possible neighborhoods of a node v of maximum degree three being part of a triangle.

Lemma 4. *Let G be a reduced graph of maximum degree three and v be a node of maximum degree that is part of a triangle (v, u_1, u_2). Then there is a branching with a branching vector of at least $(3, 3)$.*

Proof. Let u_3 be the remaining third neighbor of v. Since G is reduced, every node is part of at most one triangle, and each triangle does not contain any node of degree two. Therefore, the neighborhood of v is one of the three cases depicted in Figure 2.

Since v is of maximum degree, either v or some of its neighbors belong to some (t, k)-vertex cover, if such a cover exists. If v does not belong to any (t, k)-vertex cover, each cover C covers at least four edges with nodes from $N(v)$, because otherwise we could replace $C \cap N(v)$ with $\{v\}$. Thus, at least two nodes from $N(v)$ must be part of a (t, k)-vertex cover, if v is not.

- Let $\deg(u_3) = 1$ and C be a (t, k)-vertex cover. Without loss of generality, $u_3 \notin C$. If $\{u_1, u_2\} \subseteq C$ but $v \notin C$, $C \cup \{v\} \setminus \{u_2\}$ is a valid (t, k)-vertex cover. Thus, we can safely add v to C and no branching is necessary.
- Let $\deg(u_3) = 2$ and C be a (t, k)-vertex cover containing u_1 and u_3 but neither v nor u_2. Then $C \cup \{v\} \setminus \{u_3\}$ covers t edges with k nodes. If C contains only $\{u_1, u_2\}$, we can replace u_2 by v. Therefore, either v is part of some (t, k)-vertex cover for G, or all nodes in $\{u_1, u_2, u_3\}$ belong to such a cover, which results in a branching vector of $(3, 7)$.
- Let $\deg(u_3) = 3$. Similar to previous case, a cover containing only $\{u_1, u_2\}$ can be replaced by a cover containing $\{u_1, v\}$. Hence, either v or u_3 is part of an optimal solution, which yields the branching vector $(3, 3)$.

Lemma 5. *Let G be a graph of maximum degree three that is not three-regular and v a node of maximum degree that has a neighbor of degree two. Then there is a branching on nodes from $N[v]$ which yields a branching vector of at least $(3,3)$.*

Proof. Recall that if no (t,k)-vertex cover contains v, at least two nodes from $N(v)$ are part of any (t,k)-vertex cover, since v is of maximum degree. Let furthermore $\Delta_i := |\{\, u \in N(v) \mid \deg(u) = i \,\}|$.

- If $\Delta_1 \geq 1$ and $\Delta_3 \leq 1$, v must be part of some (t,k)-vertex cover. Since $|N(v) \cap C| \geq 2$, at least one neighbor of degree at most two must be part of some (t,k)-vertex cover. Thus, Lemma 1 implies that there exists a (t,k)-vertex cover containing all neighbors of v, and since at least one u_i is of degree one, we can replace it with v.
- Let $\Delta_1 = 1$ and $\Delta_3 = 2$. Then both neighbors of degree three are contained in some (t,k)-vertex cover, because otherwise the node with degree one would be part of some (t,k)-vertex cover. But then, this node could be replaced by v. This implies a branching vector of $(3,6)$.
- In the following we can assume that no node in $N(v)$ is of degree one. Let $\Delta_2 = 3$. Since v is not part of any (t,k)-vertex cover, Lemma 1 implies that $N(v)$ belongs to some (t,k)-vertex cover. Again, we obtain the branching vector $(3,6)$.
- Now let $1 \leq \Delta_2 \leq 2$ and $\Delta_3 = 3 - \Delta_2$. Let $u_1 \in N(v)$ be a node of degree three. Either we have directly u_1 is part of some (t,k)-vertex cover or both other neighbors. But since one of these is of degree two, Lemma 1 implies that u_1 is part of some (t,k)-vertex cover too. Therefore, the corresponding branching vector is $(3,3)$.

Combining these lemmata, we obtain a run time bound of $O^*(1.26^t)$.

Theorem 1. Partial Vertex Cover *on graphs of maximum degree three can be solved in $O^*(1.26^t)$.*

Proof. Let us first consider the case that G is a connected graph of maximum degree three. If G is not three-regular, it is easy to see that applying the branching operations or the reduction rule does not lead to a connected three-regular graph. Even more, if the branching operation splits the graph into several components, each of these components is not three-regular as well. Hence, the three-regular case can only occur at the beginning. A simple branch for each $v \in V$, where membership in the (t,k)-vertex cover is tested, increases the run time by a factor n, but leaves us with a graph that is not three-regular.

The algorithm always chooses a node of degree three with at least one neighbor of degree two or less. Thus, by Lemma 4 and Lemma 5, its branching vector is at least $(3,3)$ in t, i.e., its running time is in $O^*(1.26^t)$.

If G is not connected, let $G_0, \ldots, G_s$ be its components. For each component G_j and each $k' \leq k$ we compute the maximum number $t_{j,k'}$ of edges that can be covered in G_j with k' nodes. Each component G_j is connected, hence t calls of

the branching algorithm with parameter $0 \leq t' \leq t$ are sufficient per component. We can then use dynamic programming to compute the maximum number of edges $t_{1\cdots j,k'}$, $2 \leq j \leq n$ that can be covered with k' nodes, if only nodes from components $G_1, \ldots, G_j$ are allowed: For each $2 \leq j \leq s$ and each $0 \leq k' \leq k$, we have

$$t_{1\cdots j,k'} := \min\{\, t_{1\cdots(j-1),p} + t_{j,q} \mid p + q = k' \,\}.$$

The branching algorithm takes time $1.26^t \cdot poly(n)$, and is called $s \cdot t \cdot k$ times. Dynamic programming takes $O(s \cdot k^2)$ steps, hence we obtain an overall run time bound of $O^*(1.26^t)$ for arbitrary graphs of maximum degree three.

If we replace the actual branching with a randomized selection of the respective branching node(s), we obtain a simple randomized version of above algorithm.

Corollary 1. *There is a randomized algorithm* RPVC_3 *deciding* Partial Vertex Cover *on graphs of maximum degree three with success probability of at least* $(1/1.26)^t$.

4 A deterministic algorithm

In this section, we introduce a deterministic algorithm for arbitrary graphs. This algorithm, shown in Table 1, basically behaves as follows. A node of maximum degree is tested for membership in the (t, k)-vertex cover. If this test fails, one of its neighbors must be part of the solution, and the algorithm tests them in the decreasing order of their degrees.

Algorithm PVC(G, k, t):
Input: Graph G, k, t
select a node v of maximum degree d
let $N(v) = \{v_1, \ldots, v_d\}$ and $\deg(v_1) \geq \ldots \geq \deg(v_d)$
if $\deg(v) \leq 3$ **then** apply branching rules for graphs of maximum degree three
else for $i = 1, \ldots, d-1$ **do**
  if $i \geq \deg(v_i)$ **then return** "no"
  else if $(i < d-1)$ **and** PVC$(G - \{v\}, k-1, t-d)$ **then return** "yes"
  else return PVC$(G - \{v_{d-1}, v_d\}, k-2, t - \deg(v_{d-1}) - \deg(v_d))$

Table 1. A deterministic algorithm for Partial Vertex Cover.

Theorem 2. Partial Vertex Cover *can be solved in at most* $O^*(1.396^t)$ *steps by Algorithm* PVC.

Proof. Let G be a graph of maximum degree d. By Lemmata 2 and 3, Algorithm 1 solves Partial Vertex Cover. As for the run time, we first note that each recursive call only takes polynomial time. Now, we bound the number of recursive

calls by a function of t. To do so, we measure how t decreases in each branch and evaluate the corresponding branching vectors. If $d \leq 3$, Partial Vertex Cover can be solved in $O^*(1.26^t)$ by Theorem 1. The corresponding branching vector is $(3,3)$. If $d > 3$, either the ith recursive call in the loop returns "yes" and we obtain the branching vector

$$\big(d, \deg(v_1), \ldots, \deg(v_{i-1})\big) \trianglerighteq \big(d, i, \ldots, i\big) \trianglerighteq \big(i+1, i, \ldots, i\big).$$

Otherwise, each of these calls returns "no", so $i = d$ and we obtain the branching vector

$$\big(d, \deg(v_1), \ldots, \deg(v_{d-2}), \deg(v_{d-1}) + \deg(v_d)\big) \trianglerighteq \big(i, i, \ldots, i, i+1\big).$$

For $i \geq 5$, we may estimate the branching with the simpler branching vector

$$\big(\underbrace{i, \ldots, i}_{i \text{ times}}\big) \trianglelefteq \big(i+1, \underbrace{i, \ldots, i}_{i-1 \text{ times}}\big).$$

The characteristic polynomial of this vector is $z^i - i$ with largest positive real root $i^{1/i} \leq 5^{1/5} \leq 1.38$. For $i < 5$, we obtain the branching numbers 1.325 for the vector $(5,4,4,4)$ and 1.396 for the vector $(4,3,3)$ by a short computation. Thus, the number of recursive calls in Algorithm 1 is bounded by 1.396^t.

5 A randomized algorithm

In this section, we present a randomized algorithm for Partial Vertex Cover. Again, a node v of maximum degree is chosen deterministically, but either v or two of its neighbors are added to the (t,k)-vertex cover with certain probabilities. This technique leads to a polynomial-time algorithm with success probability of at least $1/1.2993^t$.

Fix $\alpha = (\frac{\sqrt{41}-1}{20})^{1/5} > 1/1.2993$ and let $p_d = \alpha^d$ for each $d \in \mathbf{N}$. The algorithm, see Table 2, is straight-forward and handles a only small number of border cases. We begin with an estimation that will be required later.

Lemma 6. *For $d = 4$ and $3 \leq i \leq d$, and for each $5 \leq d \in \mathbf{N}$ and each $i \in \mathbf{N}$, such that $2 \leq i \leq d$, we have*

$$(1 - \alpha^d)\binom{i}{2} \geq \alpha^{2d-2i+4}\binom{d}{2}.$$

Proof. It is sufficient to show that

$$1 \leq g(d,i) := (1-\alpha^d)\alpha^{2i-2d-4}\frac{i(i-1)}{d(d-1)}$$

for all relevant d, i. For $d = 4$ and $i \in \{3,4\}$ this is easily confirmed. Hence, fix $5 \leq d \in \mathbf{N}$. We first consider the cases $g(d,2)$ and $g(d,d)$. The function

$$h_1(z) := \frac{1}{z(z-1)}\alpha^{-2z}$$

is strictly increasing on $[5, \infty)$, since

$$\frac{d}{dz} \ln\Big(\frac{1}{z(z-1)}\alpha^{-2z}\Big) = -\frac{1}{z} - \frac{1}{z-1} - 2\ln\alpha > 0$$

for $z \geq 5$. This implies

$$h_2(z) := g(z,2) = (1-\alpha^z)\frac{2}{z(z-1)}\alpha^{-2z} = 2(1-\alpha^z)h_1(z)$$

is strictly increasing on $[5, \infty)$. Furthermore, $h_2(5) \geq 1$ — with equality for $d = 5$ — and therefore $g(d,2) \geq 1$ for each $5 \leq d \in \mathbf{N}$. Similarly, let

$$h_3(z) := g(z,z) = (1-\alpha^z)\alpha^{-4}.$$

Here, $\alpha^{-4} \approx 2.849$, and $\alpha^z \leq \frac{1}{2}$ for $z \geq 5$. Hence $h_3(z) = g(z,z) > 1$ for $z \geq 5$.

What remains to show is that $g(d,i) \geq 1$ for $i \in (2,d)$. We consider

$$f_d(z) := \ln g(d,z) = \ln\Big((1-\alpha^d)\alpha^{2z-2d-4}\frac{z(z-1)}{d(d-1)}\Big).$$

Of course $g(d,i) \geq 1$ iff $f_d(i) \geq 0$. However, $f_d(z)$ is convex on $[2,d]$, because

$$f_d''(z) = -\frac{1}{z^2} - \frac{1}{(z-1)^2} < 0.$$

With $f_d(2) = \ln g(d,2) \geq 0$ and $f_d(d) = \ln g(d,d) \geq 0$, we conclude $f_d(z) \geq 0$ on $[2,d]$ and hence $g(d,i) \geq 1$ for all $i \in \mathbf{N}$ with $2 \leq i \leq d$.

The correctness of RPVC and its success probability is due to the following lemma.

Lemma 7. *Let $G = (V,E)$ a graph.* RPVC(G,k,t) *runs in polynomial time. If there is no (t,k)-vertex cover for G, then* RPVC(G,k,t) *answers "no". Otherwise* RPVC(G,k,t) *answers "yes" with probability at least α^t.*

Proof. If there is no (t,k)-vertex cover for G, then RPVC(G,k,t) clearly answers "no". Otherwise, we use induction over the order $\succ$ on instances.

For $t = 0$, RPVC(G,k,t) answers "yes" with probability $1 = \alpha^0$. If $t > 0$ and G is not connected, let $G_0, \ldots, G_s$ be its components, and let $(k_0,t_0), \ldots, (k_s,t_s)$, such that $k_0 + \cdots + k_s = k$, $t_0 + \cdots + t_s = t$, and for each $0 \leq i \leq s$ there is a (t_i,k_i)-vertex cover for G_i. For all $0 \leq i \leq s$, we have $(G,k,t) \succ (G_i,k_i,t_i)$. Hence, by hypothesis RPVC(G_i,k_i,t_i) answers "yes" with probability at least α^{t_i}. Therefore, with probability at least

$$\alpha^{t_0}\alpha^{t_1}\cdots\alpha^{t_s} = \alpha^t$$

the dynamic programming approach is successful and RPVC(G,k,t) answers "yes". For correctness and run time of the dynamic programming approach, see the proof of Theorem 1.

Algorithm RPVC(G, k, t):
Input: Graph G, k, t
if $k < 0$ **then** return "no"
if $t \leq 0$ **then** return "yes"
if G is not connected **then**
    Compute optimal solutions for all $t' \leq t$ for every component of G.
    Combine the solutions using dynamic programming.
    Return whether there is a global solution for G.
if G has maximum degree three **then** return RPVC$_3(G, k, t)$
if G is four-regular **then**
    choose arbitrary $v \in V$
    if "yes" $\in \{\, \text{RPVC}(G - u, k - 1, t - 4) \mid u \in N[v] \,\}$ **then** return "yes"
    else return "no".
else choose $v \in V$ of maximum degree d, so that $\deg(N(v)) \neq (4, 4, 4, 4)$
$X := \binom{N(v)}{2}$
if $\deg(N(v)) = (4, 4, 4, 3)$ **then**
    $X := \{\, x \in X \mid \deg(x) = (4, 4) \,\}$
else if $\deg(N(v)) = (4, 4, 3, 3)$ **then**
    $X := \{\, x \in X \mid \deg(x) \neq (3, 3) \,\}$
Uniformly choose $C \in X$.
Return $\begin{cases} \text{RPVC}(G - v, k - 1, t - d) & \text{with probability } p_d \\ \text{RPVC}(G - C, k - 2, t - ||C||) & \text{with probability } 1 - p_d \end{cases}$

Table 2. A randomized algorithm for (t, k)-vertex cover

From now, we assume G is connected. If G has maximum degree three, by Corollary 1 RPVC$_3(G, k, t)$ answers "yes" with probability at least $(1/1.26)^t > \alpha^t$.

If G is four-regular, let v be the node chosen by the algorithm. We know that there is a solution containing at least one $u \in N[v]$. Calling RPVC$(G - u, k - 1, t - 4)$ for each $u \in N[v]$ adds factor of five to the run time. Similar to the three-regular case, it is easy to see that the respective G will be four-regular at most once on any path in the recursive call tree.

If otherwise G is not four-regular, but contains a node of degree four or larger, let v be the node of maximum degree $d > 3$ that was chosen by the algorithm. Note that it is always possible to choose v with $\deg(N(v)) \neq (4, 4, 4, 4)$, since G is not four-regular. Let C be a (t, k)-vertex cover with $v \in C$, then there is a $(k-1, t-d)$-vertex cover for $G - v$. With probability α^d the algorithm chooses v and calls RPVC$(G-v, k-1, t-d)$. Hence, by induction hypothesis, the algorithm answers "yes" with probability at least $\alpha^d \alpha^{t-d} = \alpha^t$.

If otherwise there is no (t, k)-vertex cover containing v, we know that $|C \cap N(v)| \geq 2$ for each (t, k)-vertex cover C. Fix such a (t, k)-vertex cover C for G and let $D := C \cap N(v)$, $i := |D|$, and $\deg(D) = (d_1, \ldots, d_i)$. By Lemma 1, we know $i \geq d - d_i + 2$, or $d_i \geq d - i + 2$. We begin with the two special cases that are distinguished by the algorithm:

1. If $d = 4$ and $\deg(N(v)) = (4, 4, 4, 3)$, then

$$\deg(D) \in \{\, (4, 4), (4, 4, 3), (4, 4, 4), (4, 4, 4, 3)\}.$$

With probability $(1 - \alpha^4)$ we do not choose v, and with probability at least $1/3$ we find the correct nodes $v_1, v_2 \in N(v)$. These nodes cover at least seven edges, and hence the probability to answer "yes" is, by induction, at least

$$(1 - \alpha^4)\frac{1}{3}\alpha^{t-7} = \frac{\alpha^{t-7} - \alpha^{t-4}}{3} > \alpha^t.$$

2. If $d = 4$ and $\deg(N(v)) = (4, 4, 3, 3)$, then

$$\deg(D) \in \{\, (4, 4), (4, 3, 3), (4, 4, 3, 3)\},$$

i.e., we know at least one node of degree four is in any (t, k)-vertex cover. If $i = 2$, then $\deg(D) = (4, 4)$, and we furthermore know that there is no edge between these neighbors (otherwise we can construct a (t, k)-vertex cover containing v, a contradiction). Hence, at least eight edges are being covered, and the probability to answer "yes" is, by induction, at least

$$(1 - \alpha^4)\frac{1}{5}\alpha^{t-8} = \frac{\alpha^{t-8} - \alpha^{t-4}}{5} > \alpha^t.$$

If otherwise $3 \leq i \leq 4$, we can only guarantee that six edges are being covered, but we gain an improved probability to pick two neighbors in D. We obtain a probability to answer "yes" of at least

$$(1 - \alpha^4)\frac{1}{5}\binom{i}{2}\alpha^{t-6} = \frac{\alpha^{t-6} - \alpha^{t-2}}{5}\binom{i}{2} > \alpha^t.$$

The remaining cases are $d \geq 5$, or $d = 4$ and

$$\deg(N(v)) \notin \{\, (4, 4, 3, 3), (4, 4, 4, 3), (4, 4, 4, 4)\,\}.$$

The latter enforces $i = |D| \geq 3$ due to the minimum degree d_i in D. With probability $(1 - \alpha^d)$, the algorithm does not choose v. With probability $\binom{i}{2}/\binom{d}{2}$ the algorithm chooses two correct nodes $v_1, v_2 \in D \subseteq N(v)$. Furthermore, v_1 and v_2 cover at least $2(d - i + 2)$ edges: This is clear if v_1 and v_2 are not connected or neither v_1 nor v_2 are of degree $d - i + 2$. However, if at least one node, say v_1, is of degree $d - i + 2$, and v_1 and v_2 are connected, then a (t, k)-vertex cover containing v can be constructed from C by replacing v_1 with v—a contradiction.

By induction, the probability that RPVC$(G - \{v_1, v_2\}, k - 2, t - 2(d - i + 2))$ returns "yes" is at least $\alpha^{t-2(d-i+2)}$. Therefore, the success probability of RPVC(G, k, t) is at least

$$(1 - \alpha^d)\frac{i(i-1)}{d(d-1)}\alpha^{t-2(d-i+2)} \geq \alpha^{2(d-i+2)}\alpha^{t-2(d-i+2)} = \alpha^t,$$

using the estimation from Lemma 6.

6 Exact Partial Vertex Cover

We define the parameterized problem Exact Partial Vertex Cover as follows:

Input: A graph $G = (V, E)$, positive integers k, t
Question: Is there a $C \subseteq V$, $|C| \leq k$, such that C covers *exactly* t edges?

The algorithms from the previous sections cannot be adapted to solve this problem, since there are solutions that do contain neither a node of maximum degree nor any of its neighbors.

While the first algorithm for Partial Vertex Cover by Bläser [3] might be modified to solve Exact Partial Vertex Cover as well, the random separation method by Cai, Chan, and Chan [5] solves this problem more efficiently in $O^*(2^{k+t}) = O^*(4^t)$ steps.

We use a similar and—to our best knowledge—new technique that we call *random orientation*: First randomly choose an orientation for each edge $\{v, u\}$: $v \to u$ (to u), $v \leftarrow u$ (to v), or $v - u$ (undirected). An *inner* node v is a node such that all edges incident to v are either undirected or point to v. An *inner* component U is a minimal, nonempty set U of inner nodes, such that $u \leftarrow v$ for each edge $\{u, v\}$ with $u \in U$ and $v \notin U$.

Theorem 3. *Let $G = (V, E)$ be a graph and C a solution of the* Exact Partial Vertex Cover *instance (G, k, t). Then C is a union of some inner components for a random orientation of E with probability at least 3^{-t}.*

After randomly choosing an orientation for a graph, compute all inner components. Note that no edge is incident to more than one inner component, thus we can use dynamic programming to test, whether some of these components contain together at most k nodes and cover exactly t edges. The overall run time is polynomial in t, k, and G. Obviously, the success probability is at least 3^{-t}. We easily obtain the following result:

Theorem 4. Exact Partial Vertex Cover *can be solved by a randomized algorithm in $O^*(3^t)$ with constant error probability.*

Note that this algorithm can be derandomized by using t-independent hash functions [1] yielding a run time of $O^*(c^t)$ for some constant c.

The method of random orientation can easily be used for other variants of Partial Vertex Cover, including several weighted problems. However, note that some variants are $W[1]$ hard, especially if we look for a (t, k)-vertex cover whose weight is exactly a given number (by a reduction from Subset Sum).

References

1. N. Alon, R. Yuster, and U. Zwick. Color-coding. *J. ACM*, 42(4):844–856, 1995.
2. R. Bar-Yehuda. Using homogenous weights for approximating the partial cover problem. In *Proc. of 10th SODA*, pages 71–75, 1999.

3. M. Bläser. Computing small partial coverings. *Inf. Proc. Letters*, 85:327–331, 2003.
4. N. H. Bshouty and L. Burroughs. Massaging a linear programming solution to give a 2-approximation for a generalization of the vertex cover problem. In *Proc. of 15th STACS*, number 1373 in LNCS, pages 298–308. Springer, 1998.
5. L. Cai, S. M. Chan, and S. O. Chan. Random separation: A new method for solving fixed-cardinality optimization problems. In *Proc. of 2nd IWPEC*, number 4169 in LNCS, pages 239–250. Springer, 2006.
6. J. Chen, I. A. Kanj, and G. Xia. Simplicity is beauty: Improved upper bounds for vertex cover. Technical Report TR05-008, School of CTI, DePaul University, 2005.
7. R. G. Downey and M. R. Fellows. Fixed-parameter tractability and completeness. *Congressus Numerantium*, 87:873–921, 1992.
8. F. Fomin, F. Grandoni, and D. Kratsch. A measure & conquer approach for the analysis of exact algorithms. Technical Report 359, Department of Informatics, University of Bergen, July 2007.
9. R. Gandhi, S. Khuller, and A. Srinivasan. Approximation algorithms for partial covering problems. *Journal of Algorithms*, 53:55–84, 2004.
10. J. Guo, R. Niedermeier, and S. Wernicke. Parameterized complexity of generalized vertex cover problems. In *Proc. of 9th WADS*, number 3608 in LNCS, pages 36–48, Waterloo, Canada, 2005. Springer.
11. E. Halperin and R. Srinivasan. Improved approximation algorithms for the partial vertex cover problem. In *Proc. of 5th APPROX*, number 2462 in LNCS, pages 185–199. Springer, 2002.
12. D. S. Hochbaum. The t-vertex cover problem: Extending the half integrality framework with budget constraints. In *Proc. of 1st APPROX*, number 1444 in LNCS, pages 111–122. Springer, 1998.
13. D. S. Johnson. Approximation algorithms for combinatorial problems. *J. Comput. Syst. Sci.*, 9:256–278, 1974.
14. J. Kneis, D. Mölle, S. Richter, and P. Rossmanith. Intuitive algorithms and t-vertex cover. In *Proc. of 17th ISAAC*, number 4288 in LNCS, pages 598–607. Springer, 2006.
15. J. Kneis, D. Mölle, and P. Rossmanith. Partial vs. complete domination: t-dominating set. In *Proc. of 33rd SOFSEM*, number 4362 in LNCS, pages 367–376. Springer, 2007.
16. B. Monien and E. Speckenmeyer. Ramsey numbers and an approximation algorithm for the vertex cover problem. *Acta Informatica*, 22:115–123, 1985.
17. R. Niedermeier. *Invitation to Fixed-Parameter Algorithms*. Oxford University Press, 2006.
18. J. M. Robson. Finding a maximum independent set in time $O(2^{n/4})$. Technical Report 1251-01, Université Bordeaux I, LaBRI, 2001.