

Vertex and Edge Covers with Clustering Properties: Complexity and Algorithms

Henning Fernau and David F. Manlove



UNIVERSITY
of
GLASGOW

Department of Computing Science
University of Glasgow
Glasgow G12 8QQ
UK

Technical Report
TR-2006-210
June 2006

Vertex and Edge Covers with Clustering Properties: Complexity and Algorithms

Henning Fernau^{1,*} and David F. Manlove^{2,†}

¹ *FB 4—Abteilung Informatik, Universität Trier, 54286 Trier, Germany*

² *Department of Computing Science, University of Glasgow, Glasgow G12 8QQ, UK*

Abstract

We consider the concepts of a t -total vertex cover and a t -total edge cover ($t \geq 1$), which generalize the notions of a vertex cover and an edge cover, respectively. A t -total vertex (respectively edge) cover of a connected graph G is a vertex (edge) cover S of G such that each connected component of the subgraph of G induced by S has least t vertices (edges). These definitions are motivated by combining the concepts of clustering and covering in graphs. Moreover they yield a spectrum of parameters that essentially range from a vertex cover to a connected vertex cover (in the vertex case) and from an edge cover to a spanning tree (in the edge case). For various values of t , we present \mathcal{NP} -completeness and approximability results (both upper and lower bounds) and \mathcal{FPT} algorithms for problems concerned with finding the minimum size of a t -total vertex cover, t -total edge cover and connected vertex cover, in particular improving on a previous \mathcal{FPT} algorithm for the latter problem.

1 Introduction

In graph theory, the notion of covering vertices or edges of graphs by other vertices or edges has been extensively studied (see [26] for a survey). For instance, covering vertices by other vertices leads to parameters concerned with vertex domination [21, 22]. When edges are to be covered by vertices we obtain parameters connected with the classical vertex covering problem [20, p.94]. Covering vertices by edges, i.e. finding edge covers, was first considered by Norman and Rabin [32]. Finally, when edges are to cover other edges, we obtain parameters associated with edge domination (introduced by Mitchell and Hedetniemi [27]). These problems have long been a testbed for the design of parameterized algorithms (or for showing the limitations of that approach) [11]. However, in particular only recently has a systematic study of variants of vertex cover problems been initiated with respect to parameterized complexity [19, 31].

Clustering in graphs is another fundamental concept with a large range of practical applications [15]. Connectedness can be seen as one of the weakest notions of clustering: it is reasonable to assert that a vertex set can be termed a cluster only if it is connected. When being used for classification purposes, there is rarely only one cluster, but rather a number of them, each representing some concept, i.e., one is looking for connected components. In order to exclude trivial cases and to define meaningful concepts, it may often be appropriate to impose a lower bound on the number of elements per cluster.

*Part of this work was carried out whilst visiting the University of Glasgow, supported by EPSRC grant EP/D030110/1. Email henningfernau@yahoo.de.

†Supported by EPSRC grant GR/R84597/01 and RSE / Scottish Executive Personal Research Fellowship. Email davidm@dcs.gla.ac.uk.

In this paper we consider a synergy of the notion of clustering with each of the concepts of vertex and edge covering. Throughout we assume that $G = (V, E)$ is a connected graph, where $n = |V|$ and $m = |E| \geq 1$. For $1 \leq t \leq n$, a t -total vertex cover (henceforth a t -tvc) in G is a vertex cover S in G such that each connected component of $G[S]$, the subgraph of G induced by S , has at least t vertices. Similarly, for $1 \leq t \leq m$, a t -total edge cover (henceforth a t -tec) in G is an edge cover S of G (i.e. each vertex of G is incident to an edge in S) such that each connected component of $G[S]$, the subgraph of G induced by S , has at least t edges. Hence, if S is a t -tvc or t -tec, then S is a vertex cover or edge cover respectively such that each member of S belongs to a “cluster” containing at least t elements of S .

The concept of a *total dominating set* in a graph, first defined and studied by Cockayne et al. [6], illustrates one case where the notions of clustering and covering (vertices by vertices) have already been brought together. A set of vertices S is a *total dominating set* of G if (i) S is a dominating set (i.e. every vertex in $V \setminus S$ is adjacent to a vertex in S), and (ii) each connected component of $G[S]$ has at least two vertices.

The notion of a 2-tvc was first defined by Jean Blair [2] using the terminology *total vertex cover* (by analogy to the term *total dominating set*). It is straightforward to present relationships between the minimum size of a t -tvc (respectively t -tec) for various values of t and established parameters concerned with vertex covering (respectively edge covering) in G . Throughout this paper, our notation follows and extends that of Harary [20]. Let $\alpha_0(G)$ denote the minimum size of a vertex cover in G . A *connected vertex cover* (henceforth a *cvc*) in G is a vertex cover S in G such that $G[S]$ is connected. Let $\alpha_0^c(G)$ denote the minimum size of a cvc in G . It follows that a 1-tvc is simply a vertex cover (recall that $m \geq 1$), whilst a cvc of size t is a t -tvc. For $t \geq 1$, let $\alpha_{0,t}(G)$ denote the minimum size of a t -tvc in G . Then $\alpha_{0,1}(G) = \alpha_0(G)$. The parameters $\alpha_{0,t}(G)$ for $t \geq 2$ do not appear to have been studied in the literature previously. In Section 2, we present some additional relationships involving the parameters $\alpha_0(G)$, $\alpha_{0,t}(G)$ and $\alpha_0^c(G)$.

Now let $1 \leq t \leq m$ – we turn to the concept of a t -tec. It follows that a 1-tec is simply an edge cover (again recall that $m \geq 1$), whilst a minimum $(n - 1)$ -tec is a minimum connected edge cover, i.e. a spanning tree. Let $\alpha_{1,t}(G)$ denote the minimum size of a t -tec of G , and let $\alpha_1(G)$ denote the minimum size of an edge cover of G . Then $\alpha_{1,1}(G) = \alpha_1(G)$. The parameters $\alpha_{1,t}(G)$ for $t \geq 2$ do not appear to have been studied in the literature previously. In Section 2, we present some additional relationships between the parameters $\alpha_1(G)$ and $\alpha_{1,t}(G)$.

We remark that, for a t -tvc (respectively t -tec) to exist, it is sufficient that each connected component of G has at least t vertices (edges), and the results in this paper also hold in such a setting. However for ease of exposition, and due to the correspondence between t -tvcs and t -tecs with connected vertex covers and spanning trees respectively, we choose to assert throughout that G is connected.

Given $t \geq 1$, let VC, t -TVC, t -TEC and CVC denote the problems of computing $\alpha_0(G)$, $\alpha_{0,t}(G)$, $\alpha_{1,t}(G)$ and $\alpha_0^c(G)$ respectively, given a connected graph G where $n = |V|$ and $m = |E| \geq 1$ (additionally $n \geq t$ in the case of t -TVC and $m \geq t$ in the case of t -TEC). Let VC-D, t -TVC-D, t -TEC-D and CVC-D denote the decision versions of VC, t -TVC, t -TEC and CVC, respectively. Hence, the question is, given a graph G and a parameter k , whether there is a cover C (with the additional properties specified by the problem) such that $|C| \leq k$.

For each $t \geq 2$, we show in Section 3 that t -TVC is \mathcal{NP} -hard and not approximable within an asymptotic performance ratio of $10\sqrt{5} - 21 - \delta$ (> 1.3606), for any $\delta > 0$, unless $\mathcal{P} = \mathcal{NP}$. However on the other hand we prove that t -TVC is approximable within 2. We also prove that t -TVC-D is \mathcal{NP} -complete, even for planar bipartite graphs of maximum degree

3. Moreover we show that there exists a constant $\delta_t > 1$ such that t -TVC in bipartite graphs of maximum degree 3 is not approximable within δ_t unless $\mathcal{P} = \mathcal{NP}$. Finally, we give a parameterized algorithm for 2-TVC-D with complexity $\mathcal{O}^*(2.3655^k)$. Here the parameter is the size of the 2-tvc.

CVC is \mathcal{NP} -hard, even for planar graphs of maximum degree 4 [17], though polynomial-time solvable for graphs of maximum degree 3 [36]. For a tree T , finding a minimum cvc is trivial (if $T = K_2$, one vertex will suffice, otherwise the set of non-leaf nodes is a minimum cvc in T). It is known that CVC is approximable within 2 [35, 1]. In Section 4, we show that CVC is not approximable within an asymptotic performance ratio of $10\sqrt{5} - 21 - \delta$, for any $\delta > 0$, unless $\mathcal{P} = \mathcal{NP}$. The complexity of CVC in bipartite graphs does not seem to have been considered in the literature so far. We show that CVC-D is \mathcal{NP} -complete, even for planar bipartite graphs of maximum degree 4. We also present a parameterized algorithm for CVC-D with complexity $\mathcal{O}^*(2.9316^k)$, improving on a previous algorithm due to Guo et al. [19], having complexity $\mathcal{O}^*(6^k)$. Here the parameter is the size of the cvc. We remark that, independently and by using different techniques, Moelle et al. [28] present a parameterized algorithm for CVC-D with complexity $\mathcal{O}^*(3.2361^k)$. Furthermore, following that approach, an improved algorithm for the same problem, having complexity $\mathcal{O}^*(2.7606^k)$, will be reported [29].

1-TEC, i.e. the problem of finding a minimum edge cover, is polynomial-time solvable [32]. In Section 5, we give a Gallai identity involving $\alpha_{1,t}(G)$ for each $t \geq 1$. We use this to prove that t -TEC-D is \mathcal{NP} -complete for each $t \geq 2$. We also show that t -TEC is approximable within 2 for each $t \geq 2$, though there exists some $\delta > 1$ such that 2-TEC is not approximable within δ unless $\mathcal{P} = \mathcal{NP}$. Finally we show that t -TEC-D is in \mathcal{FPT} for each $t \geq 2$ (where the parameter is the size of the t -tec) and the parametric dual of 2-TEC-D is also in \mathcal{FPT} . This gives one of the few examples where both a problem and its dual belong to \mathcal{FPT} .

2 Preliminary observations involving $\alpha_{0,t}(G)$ and $\alpha_{1,t}(G)$

We begin this section by presenting some relationships involving the parameters $\alpha_0(G)$, $\alpha_{0,t}(G)$ and $\alpha_0^c(G)$.

Proposition 1. *Let $G = (V, E)$ be a connected graph where $n = |V|$, $m = |E| \geq 1$, and let $1 \leq t \leq n$. Then:*

1. $\alpha_0(G) \leq \alpha_{0,t}(G)$, and for $t < n$, $\alpha_{0,t}(G) \leq \alpha_{0,t+1}(G)$;
2. $\alpha_{0,t}(G) \geq t$;
3. for $\alpha_0^c(G)/2 < t \leq \alpha_0^c(G)$, $\alpha_{0,t}(G) = \alpha_0^c(G)$;
4. for $t \geq \alpha_0^c(G)$, $\alpha_{0,t}(G) = t$;
5. the minimum t such that $\alpha_{0,t}(G) = t$ satisfies $t = \alpha_0^c(G)$.

Proof. 1. If S is a $(t+1)$ -tvc then clearly S is a t -tvc. Moreover clearly any t -tvc is a vertex cover.

2. If S is any t -tvc, then as $m \geq 1$, it follows that $G[S]$ has at least one connected component, which contains at least t vertices.

3. Let S be a minimum t -tvc and let C be a minimum cvc. Then C is a t -tvc, so that $|S| \leq |C| = \alpha_0^c(G)$. Now suppose that $G[S]$ contains at least two connected components. Then $|S| \geq 2t > \alpha_0^c(G)$, a contradiction. Hence S is a cvc, so that $|C| \leq |S|$. Hence $\alpha_{0,t}(G) = \alpha_0^c(G)$.

4. Let C be a minimum cvc and let $t' = t - |C|$. As G is connected we may construct a t -tvc S by adding t' vertices to C . Then $|S| = t$, so that $\alpha_{0,t}(G) \leq t$. Hence $\alpha_{0,t}(G) = t$ by Part 2.
5. Let $t = \alpha_0^c(G)$. By Part 4, $\alpha_{0,t}(G) = t$. Now suppose that $t' < t$ and $\alpha_{0,t'}(G) = t'$. Let S be a t' -tvc such that $|S| = t'$. Then $G[S]$ contains one connected component, for otherwise $|S| \geq 2t'$, a contradiction. Hence S is a cvc such that $|S| = t' < \alpha_0^c(G)$, a contradiction. \square

We next present some relationships involving the parameters $\alpha_1(G)$ and $\alpha_{1,t}(G)$.

Proposition 2. *Let $G = (V, E)$ be a connected graph where $n = |V|$, $m = |E| \geq 1$, and let $1 \leq t \leq m$. Then:*

1. $\alpha_1(G) \leq \alpha_{1,t}(G)$, and for $t < m - 1$, $\alpha_{1,t}(G) \leq \alpha_{1,t+1}(G)$;
2. $\alpha_{1,t}(G) \geq t$;
3. for $\frac{n-1}{2} < t \leq n - 1$, $\alpha_{1,t}(G) = n - 1$;
4. for $t \geq n - 1$, $\alpha_{1,t}(G) = t$.
5. the minimum t such that $\alpha_{1,t}(G) = t$ satisfies $t = n - 1$.

Proof. 1. If S is a $(t + 1)$ -tec then clearly S is a t -tec. Moreover clearly any t -tec is an edge cover.

2. If S is any t -tec, then as $m \geq 1$, it follows that $G[S]$ has at least one connected component, which contains at least t edges.
3. Let S be a minimum t -tec and let T be a spanning tree of G . Then T is a t -tec, so that $|S| \leq |T| = n - 1$. Now suppose that $G[S]$ contains at least two connected components. Then $|S| \geq 2t > n - 1$, a contradiction. Hence $G[S]$ is connected, so that $|S| \geq n - 1$. Thus $\alpha_{1,t}(G) = n - 1$.
4. Let T be a spanning tree of G and let $t' = t - (n - 1)$. As G is connected we may construct a t -tec S by adding t' edges to T . Then $|S| = t$, so that $\alpha_{1,t}(G) \leq t$. Hence $\alpha_{1,t}(G) = t$ by Part 2.
5. Let $t = n - 1$. By Part 4, $\alpha_{1,t}(G) = t$. Now suppose that $t' < t$ and $\alpha_{1,t'}(G) = t'$. Let S be a t' -tec such that $|S| = t'$. Then $G[S]$ contains one connected component, for otherwise $|S| \geq 2t'$, a contradiction. Hence S is a spanning tree such that $|S| = t' < n - 1$, a contradiction. \square

3 Complexity and approximability of t -TVC

We begin with a lower bound for the approximability of t -TVC in general graphs.

Theorem 3. *For each $t \geq 1$, t -TVC is \mathcal{NP} -hard and not approximable within an asymptotic performance ratio of $10\sqrt{5} - 21 - \delta$, for any $\delta > 0$, unless $\mathcal{P} = \mathcal{NP}$.*

Proof. For $t = 1$ the result follows by [8]. Now assume that $t \geq 2$. Let $G = (V, E)$ be an instance of vc. We lose no generality in assuming that G is connected and $|V| \geq 2$. Create a new graph $G' = (V', E')$ such that $V' = V \cup W$ and $E' = E \cup E_1 \cup E_2$, where $W = \{w_i : 1 \leq i \leq t\}$ is a set of new vertices, $E_1 = \{\{v, w_1\} : v \in V\}$ and $E_2 = \{\{w_i, w_{i+1}\} : 1 \leq i \leq t - 1\}$. Let $W' = W \setminus \{w_t\}$. It is straightforward to verify that

if S is a minimum vertex cover in G , then $S \cup W'$ is a t -tvc in G' . Conversely if S' is a minimum t -tvc in G' , then $S' \cap W = W'$, and $S' \cap V$ is a vertex cover in G . Hence $\alpha_{0,t}(G') = \alpha_0(G) + t - 1$. The result follows by [8]. \square

We now present an upper bound for the approximability of t -TVC.

Theorem 4. *For each $t \geq 1$, t -TVC is approximable within 2.*

Proof. Let $G = (V, E)$ be an instance of t -TVC (then G is a connected graph, where $n = |V| \geq t$ and $m = |E| \geq 1$). Savage [35] presents an approximation algorithm for CVC: the algorithm computes a cvc S in G such that $|S| \leq 2\alpha_0(G)$. Suppose firstly that $t \leq |S|$. Then S is a t -tvc, and $|S| \leq 2\alpha_0(G) \leq 2\alpha_{0,t}(G)$ by Proposition 1, as required. Now suppose that $t > |S|$. Let $t' = t - |S|$. As G is connected, we may construct a t -tvc S' in G by adding t' vertices to S . Then $|S'| = t$, so that S' is in fact a minimum t -tvc by Proposition 1. \square

The next two results concern the complexity and approximability of t -TVC in bounded degree bipartite graphs, for each $t \geq 2$.

Theorem 5. *For each $t \geq 2$, t -TVC-D is \mathcal{NP} -complete for planar bipartite graphs of maximum degree 3.*

Proof. Clearly t -TVC-D belongs to \mathcal{NP} . To show \mathcal{NP} -hardness, we give a reduction from the \mathcal{NP} -complete restriction of VC-D to planar graphs of maximum degree 3 [18, 17]. Hence let $G = (V, E)$ (a planar graph of maximum degree 3) and k (a positive integer) be an instance of this problem. Let $E = \{e_1, e_2, \dots, e_m\}$ for some m . We define an instance of t -TVC-D as follows. Construct a graph $G' = (V', E')$ by letting $V' = V \cup W$, where $W = \{w_{i,j} : 1 \leq i \leq m \wedge 1 \leq j \leq t\}$. For each i ($1 \leq i \leq m$), suppose that $e_i = \{u, v\}$ for some $u, v \in V$. Add the edges $\{u, w_{i,1}\}$, $\{w_{i,j}, w_{i,j+1}\}$ ($1 \leq j \leq t-1$) and $\{w_{i,1}, v\}$ to E' . Clearly G' can be constructed in polynomial time from G , and G' is planar, bipartite and has maximum degree 3. Let $k' = k + (t-1)m$. We claim that G has a vertex cover of size at most k if and only if G' has a t -tvc of size at most k' .

For, suppose that G has a vertex cover S of size at most k . Let $S' = S \cup W'$, where $W' = W \setminus \{w_{i,t} : 1 \leq i \leq m\}$. Then it may be verified that S' is a t -tvc of G' , and $|S'| = |S| + (t-1)m \leq k + (t-1)m = k'$.

Conversely suppose that G' has a t -tvc of size at most k' . Choose S' to be such a set that minimizes $|S' \cap W|$. It is straightforward to verify that $W' \subseteq S'$, since $t \geq 2$. Also, $S' \cap W = W'$. For, suppose that $w_{i,t} \in S'$ for some i ($1 \leq i \leq m$). Let $e_i = \{u, v\}$ for some $u, v \in V$. Define $S'' = (S' \setminus \{w_{i,t}\}) \cup \{u\}$. Then S'' is a t -tvc of G' , $|S''| \leq |S'| \leq k'$, and $|S'' \cap W| < |S' \cap W|$, contradicting the choice of S' . Hence the claim is established. Let $S = S' \cap V$. Then it may be verified that S is a vertex cover of G , and $|S| = |S'| - (t-1)m \leq k' - (t-1)m = k$. \square

Corollary 6. *For each $t \geq 2$, t -TVC in bipartite graphs of maximum degree 3 is not approximable within $1 + \frac{1}{500t-400}$ unless $\mathcal{P} = \mathcal{NP}$.*

Proof. VC in cubic graphs is not approximable within $\frac{100}{99}$ unless $\mathcal{P} = \mathcal{NP}$ [5]. By considering this problem as the starting point for the same reduction as in the proof of Theorem 5, it again follows that $\alpha_{0,t}(G') = \alpha_0(G) + (t-1)m$. Now $\alpha_0(G) \geq \beta_1(G) \geq \frac{m}{5}$ [37, Theorem 60], where $\beta_1(G)$ is the size of a maximum matching in G , since G is cubic. It follows that $\alpha_{0,t}(G') \leq (5t-4)\alpha_0(G)$. Hence the reduction of Theorem 5 is an L-reduction (defined in [33]) with parameters $\alpha = 5t-4$ and $\beta = 1$. The result follows by [37, Theorem 63]. \square

We next present a result concerning the parameterized complexity of 2-TVC.

Theorem 7. *2-TVC-D is in FPT and can be solved in time $\mathcal{O}^*(2.3655^k)$, where k is the size of the 2-tvc.*

Proof. Let $G = (V, E)$ be a connected graph. Firstly, we describe an algorithm running in time $\mathcal{O}^*(4^k)$. It is known (see [11, 7]) that all minimal vertex covers of size at most k can be enumerated in time $\mathcal{O}^*(2^k)$. Of course, a valid (minimal) vertex cover need not be a valid 2-tvc, but all 2-tvcs of size k can be obtained by “extending” minimal vertex covers of size at most k .

For each minimal vertex cover C of G described at a leaf of the search tree, we construct a hypergraph H as follows: $V' = V \setminus C$ are the vertices of the hypergraph, and the hyperedges E' are the open neighbourhoods of the vertices in C that do not contain (other) vertices from C . Now, a “minimum extension” of C to a valid 2-tvc corresponds to a minimum hitting set in H . Since $|E'| \leq k$, this can be done in time $\mathcal{O}^*(2^k)$ according to [13]; see also [11, Theorem 8.1]. This shows that 2-TVC-D can be solved in time $\mathcal{O}^*(4^k)$.

To improve on this running time, observe that the degree-0 and degree-1 reduction rules of VC-D (c.f. [11, p.21]) are also valid for 2-TVC-D with some variation, since we should now deal with instances in which some vertices are already marked. More precisely, we say that a vertex is 1-marked if it is known to belong to the vertex cover, and it is 2-marked if, in addition, also one of its neighbours is known to belong to the vertex cover. Moreover, a vertex is 0-marked if it is unknown if it belongs to the vertex cover, but one of its neighbours does belong to the vertex cover. Notice that the particular neighbour that testifies why a vertex has become say 0-marked may be later deleted since the necessary information is kept in the markings. To be coherent with the search tree part, we manage the parameter budget in a way that the 1- and 2-marked vertices are already taken into account; hence we have a NO-instance (corresponding to this branch of the search tree) if the parameter budget falls below 0.

We now introduce two budgets (parameters) k_1 and k_2 , both bounded by the original parameter size of k in the beginning, where k_1 actually bounds the first part (the vertex cover enumeration, although this dictum is not strictly true) and k_2 bounds the second part (the Hitting Set phase) of the search tree. Obviously, $k_1 \leq k_2$ as long as we are in the vertex cover enumeration phase.

We use the following *colouring handling rules*:

- If vertex x is unmarked but neighbour of a 1-marked or 2-marked vertex, then 0-mark x .
- If vertex x is 1-marked and neighbour of a 1-marked or 2-marked vertex, then 2-mark x .
- Merge two 2-marked vertices and decrement k_2 .
- If the parameter k_1 drops below 0, we have a NO-instance.

These rules guarantee that the 1-marked and 2-marked vertices form an independent set in the graph, and that there is at most one 2-marked vertex in a reduced instance.

In the following, we always assume that we have already exhaustively executed the colouring handling rules.

Let us first consider the case that x is a vertex of degree 0.

- If x is unmarked or 0-marked, then delete x .
- If x is 1-marked, then we have a NO-instance.
- If x is 2-marked, then delete x .

If x is a vertex of degree 1 with unique neighbour y , then do the following:

- If x is unmarked or 0-marked, we distinguish two subcases:
 - If y has degree 1, decrease both parameters k_1 and k_2 by 2 if x and y are both unmarked, and decrease both parameters k_1 and k_2 by 1 if x or y is 0-marked. In both cases, delete both x and y .
 - Otherwise, y has degree at least 2.
 - * If y is unmarked, 1-mark y , delete x and decrement k_1 .
 - * If y is 0-marked, 2-mark y , delete x and decrement k_1 .
- (For if C is a minimum 2-tvc that includes x , then $(C \setminus \{x\}) \cup \{z\}$ is a minimum 2-tvc excluding x , where $z \neq x$ is adjacent to y .)
- If x is 1-marked, then 2-mark y and decrement the parameter k_1 .
 - Finally, if x is 2-marked, then delete x .
- Notice: y has been already 0-marked by the colouring rules.

If the vertices that are unmarked or 0-marked form an independent set, a vertex cover has been found; without any 1-marked vertices, we have a 2-tvc.

Let us look at a first simple example, namely *triangles*. Hence, assume we have three vertices x, y, z in the graph that are mutually neighboured. Assume furthermore that all of them are either unmarked or 0-marked. Already for the classical vertex cover case, (i.e., 1-TVC-D), we know that two out of these three vertices must go into the cover. Notice that in this special case these two vertices will be neighbours, so that this branching scenario is also valid for 2-TVC-D, since clusters are automatically created. Therefore, the vertices put into the cover by this branching need not be taken care of later in the Hitting Set phase. Hence, the budget for that phase can be reduced by 2; in fact, this would be done by the colouring rule that merges two 2-marked vertices. The corresponding recurrence is therefore:

$$T(k_1, k_2) \leq 3T(k_1 - 2, k_2 - 2)$$

Fortunately, we already know the complexity of the second phase, which is $\mathcal{O}^*(2^{k_2})$. Assuming a running time of $\mathcal{O}^*(c^{k_1})$ for the first phase (with c still to be determined), we obtain the condition

$$c^{k_1} 2^{k_2} \leq 3c^{k_1-2} 2^{k_2-2}.$$

Multiplication with $c^{-k_1+2} 2^{-k_2+2}$ yields: $4c^2 \leq 3$, i.e., $c \leq \sqrt{3}/2$. The overall running time of the algorithm can be grossly estimated by assuming $k_1, k_2 \leq k$, so that in that particular case $(2 \cdot c)^k \leq \sqrt{3}^k \leq 1.7321^k$ follows.

Let us now consider another special case with a very nice branching behaviour: that of two neighboured vertices x, y that are 0-marked. Clearly, either x or y must go into the cover in order to cover the edge xy . Whichever vertex we put into the cover, notice that it will be immediately 2-marked (and hence it will not be considered in the Hitting Set phase). Therefore, also the parameter k_2 is decreased. We are led to the recurrence:

$$c^{k_1} 2^{k_2} \leq 2c^{k_1-1} 2^{k_2-1}$$

which is obviously solved by $c \leq 1$, yielding $(2 \cdot c)^k \leq 2^k$. Notice that when we have a triangle with one of its vertices 1-marked or 2-marked, then the other two vertices will be 0-marked (assuming a reduced instance), and hence we may assume in the following that our graph contains no triangles (with whatever marking).

After this type of branching, all neighbours of 0-marked vertices are unmarked or 1-marked or 2-marked. Notice that we will enter the Hitting Set phase when there are no more unmarked vertices around. Hence, we can assume in the following that we can always find an unmarked vertex y that is neighbour of a 0-marked vertex x . We will branch at y or its neighbours according to what we describe in the following. In the corresponding analysis, we often use the idea that when x is put into the cover, then (since x is 0-marked) also the second parameter is decreased by 1 (actually, it would be possible to decrease the second parameter even by 2 if we knew that x is neighbour of a 1-marked vertex, but the worst case is that all marked neighbours of x are 2-marked).

If $\deg(y) \geq 7$, we simply branch at y : either y is put into the cover (decrementing only k_1) or all its at least 7 neighbours are put into the cover. The latter branch decreases k_1 by 7 and decrements k_2 , since x is put into the cover. We are led to the recurrence:

$$c^{k_1} 2^{k_2} \leq c^{k_1-1} 2^{k_2} + c^{k_1-7} 2^{k_2-1}$$

which is equivalent to $2c^7 \leq 2c^6 + 1$; finally yielding $(2 \cdot c)^k \leq 2.3653^k$. This will be (nearly) our worst-case scenario in the end.

If $\deg(y) \leq 6$, we use another good branching idea that is indeed rather special to the cluster concept in 2-TVC-D: If v is a 1-marked vertex in a reduced instance, then one of its neighbours must be put into the cover in order to satisfy the cluster condition. For small-degree 1-marked vertices, this leads to a very satisfactory branching behaviour. We will use this idea in the subcase that takes y into the cover.

Consider the case $\deg(y) = 6$. Let $N(y) = \{x, z_1, z_2, z_3, z_4, z_5\}$ describe the neighbourhood of y . If we take y into the cover, then one $v \in N(y)$ must go into the cover, since y will then become 1-marked.

- If z_1 is put into the cover, then both k_1 and k_2 are reduced by 2.
- If z_1 is not put into the cover, $N(z_1)$ must be part of the cover. Assume that now z_2 is put into the cover.
- Otherwise, both z_1 and z_2 are not put into the cover but their neighbourhoods are. Assume also that z_3 goes into the cover.
- This argument continues up to the point that none of the z_i are put into the cover but all their neighbours are. In addition, x is put into the cover.

If we do not take y into the cover, then (as before) all neighbours of y are put into the cover, in particular x , so that the second parameter is reduced by 1.

To find a good estimate for the recursion, we have to reason about possible common vertices and minimum degrees. First of all, as we shall see later, we may assume that all vertices z_i have each at least three neighbours. We can also assume that for all i, j : $z_i \notin N(z_j)$ and that $x \notin N(z_j)$, since this would mean triangles in our instance. So the neighbourhood of $\{z_i, z_{i+1}\}$ or $\{z_i, x\}$ will consist of at least three vertices. All these vertices plus y will be put into the cover in all but the first case of the case distinction above, therefore reducing the first parameter by 4 and the second by 2 (at least). We get as an overall estimate for the recursion:

$$c^{k_1} 2^{k_2} \leq c^{k_1-2} 2^{k_2-2} + 5c^{k_1-4} 2^{k_2-2} + c^{k_1-6} 2^{k_2-1}$$

The first term of the right-hand side comes from the case that puts both y and z_1 into the cover, and the last term represents the case that y is not put into the cover but all its neighbours are. A little algebra reveals that $(2c)^k \leq 2.3655^k$ in this case. In fact, this is the overall worst case estimate of our algorithm.

Similarly, in the case $\deg(y) = 5$, we can derive

$$c^{k_1} 2^{k_2} \leq c^{k_1-2} 2^{k_2-2} + 4c^{k_1-4} 2^{k_2-2} + c^{k_1-5} 2^{k_2-1}$$

leading to $(2c)^k \leq 2.3055^k$. Assuming $\deg(y) = 4$, we get:

$$c^{k_1} 2^{k_2} \leq c^{k_1-2} 2^{k_2-2} + 3c^{k_1-4} 2^{k_2-2} + c^{k_1-4} 2^{k_2-1}$$

and hence $(2c)^k \leq 2.2361^k$. Similarly, in the case $\deg(y) = 3$, we can derive

$$c^{k_1} 2^{k_2} \leq c^{k_1-2} 2^{k_2-2} + 2c^{k_1-4} 2^{k_2-2} + c^{k_1-3} 2^{k_2-1}$$

leading to $(2c)^k \leq 2.1454^k$.

We now consider the case that $\deg(y) \geq 3$ and that y has a neighbour $z_1 \neq x$ of degree 2 (already excluding the case of degree 1 due to reduction rules). Then, we can branch as follows: either take y into the cover or take all of $N(y)$. The analysis of this branching depends on whether or not z_1 is unmarked. If z_1 is unmarked, then not taking y into the cover not only puts the (three or more) neighbours of y into the cover, but it will (due to the fact that y is deleted from the instance) be the case that z_1 is then of degree 1; hence, a reduction rule will trigger in the recursive call of the procedure and then the unique neighbour of z_1 will be put into the cover as well, in order to satisfy the cluster condition. Therefore, we get the following overall recurrence:

$$c^{k_1} 2^{k_2} \leq c^{k_1-1} 2^{k_2} + c^{k_1-4} 2^{k_2-3}$$

This leads us to $(2c)^k \leq 2.1903^k$. If however z_1 is 0-marked, then in the case that y is not put into the cover at least two 0-marked vertices will be 2-marked (and therefore they disappear by reduction rules in the next recursive call). This leads to the following recurrence:

$$c^{k_1} 2^{k_2} \leq c^{k_1-1} 2^{k_2} + c^{k_1-3} 2^{k_2-2}$$

which means that $(2c)^k \leq 2.3594^k$.

Finally, we consider the remaining case that $\deg(y) = 2$, i.e., $N(y) = \{x, z\}$. If we choose y , then we need to choose either x or z to satisfy the cluster property, since y is unmarked. If we do not choose y , then we need to choose both x and z . When choosing x , then x will become 2-marked. This leads to the following recurrence:

$$c^{k_1} 2^{k_2} \leq 2c^{k_1-2} 2^{k_2-2} + c^{k_1-2} 2^{k_2-1}.$$

Thus $(2c)^k \leq 2^k$. This concludes our case discussion.

In fact, the algorithm we analyzed can be written up in quite a compact form (see Alg. 1—here, the pseudocode given by Alg. 2 is essentially a macro for the operation “branch at y ” which should be inserted in place of these operations at the relevant points in Alg. 1; -1 encodes unmarked vertices) The reader may verify that the reduction rules will always lead to the algorithm behaviour as analyzed above. Notice that the third heuristic priority for branching will be used only once in the very beginning of the algorithm for each component. However, we can simply modify the algorithm so that it actually always computes a minimum 2-tvc (not only some 2-tvc satisfying the bound k), and we can compute the 2-tvc componentwisely; the very first component will determine the run time estimate, since for all other components the parameter is already reduced. The correctness of the algorithm is immediate from what is said before. \square

Algorithm 1 An advanced search tree algorithm for 2-TVC-D, called TVC-ST

Require: an annotated graph $G = (V, E)$ with marking functions $\mu : V \rightarrow \{-1, 0, 1, 2\}$, a positive integer k

Ensure: return C if G has a 2-tvc set $C \subseteq V$ with $|C| \leq k$; NO otherwise

$C' := \emptyset$; $\{C'$ collects the “new vertices” in the cover}
exhaustively apply the reduction rules; $\{\text{this might also modify } C'\}$
 $\{\text{the resulting instance will be also called } (G = (V, E), \mu, k) \text{ as before}\}$
if $k < 0$ **then**
5: return NO;
 else if there are two neighboured vertices x, y with $\mu(x) = \mu(y) = 0$ **then**
 branch at y ;
 else if G contains no more unmarked vertices (i.e., no x with $\mu(x) = -1$) **then**
 form the corresponding Hitting Set instance with the 0-marked vertices as vertices of the hypergraph and the hyperedges $N(x)$ for x being 1-marked;
10: compute a minimum hitting set H ;
 if $|H| > k$ **then**
 return NO;
 else
 return $H \cup C'$;
15: **end if**
 else
 select an unmarked vertex y for branching according to the following list of priorities:
 1. if possible y should belong to a triangle
 2. if possible a vertex x with $\mu(x) = 0$ that has a neighbour y that is unmarked
 3. otherwise, select any unmarked y
 branch at y ;
 end if

Algorithm 2 The code of “branch at y ”

modify μ by setting $\mu(y) = 1$;
 $C := \text{TVC-ST}(G, \mu, k - 1)$;
if $C \neq \text{NO}$ **then**
 return $C \cup \{y\} \cup C'$;
5: **else**
 modify μ by setting $\mu(v) := \mu(v) + 2$ for all $v \in N(y)$ with $\mu(v) < 1$ at present;
 delete y from G (and from μ);
 $C := \text{TVC-ST}(G, \mu, k - \text{deg}(y))$;
 if $C \neq \text{NO}$ **then**
10: return $C \cup N(y) \cup C'$;
 else
 return NO;
 end if
 end if
end if

4 Complexity and approximability of CVC

We begin with two results concerning the complexity and approximability of CVC in general graphs and planar bipartite graphs of bounded degree.

Theorem 8. *CVC is not approximable within an asymptotic performance ratio of $10\sqrt{5} - 21 - \delta$, for any $\delta > 0$, unless $\mathcal{P} = \mathcal{NP}$.*

Proof. The result follows using the construction in the proof of Theorem 3 for the case that $t = 2$. \square

Theorem 9. *CVC-D is \mathcal{NP} -complete for planar bipartite graphs of maximum degree 4.*

Proof. Clearly CVC-D belongs to \mathcal{NP} . To show \mathcal{NP} -hardness, we use the same reduction as in Theorem 5 with $t = 2$, however in this case we reduce from the \mathcal{NP} -complete restriction of CVC-D to planar graphs of maximum degree 4 [17]. The graph G' so constructed is then also a planar bipartite graph of maximum degree 4. If S is a connected vertex cover of size at most k in G , then $S \cup W'$ is a connected vertex cover of size at most k' in G' . Conversely if S' is a minimum connected vertex cover of size at most k' in G' , then $S' \cap W = W'$. It follows that $S' \cap V$ is a connected vertex cover in G of size at most k . \square

We now show how to use the colouring techniques from the proof of Theorem 7 in order to give a parameterized algorithm for CVC-D that improves on the previous $\mathcal{O}^*(6^k)$ algorithm described in [19].

Theorem 10. *CVC-D is in FPT and can be solved in time $\mathcal{O}^*(2.9316^k)$, where k is the size of the cvc.*

Proof. The algorithm proceeds along the lines of the one suggested in [19], using two stages: firstly, we enumerate all minimal vertex covers of size at most k , and secondly we apply an algorithm for solving the STEINER TREE PROBLEM on (at most) k terminals. Using the $\mathcal{O}^*(2^k)$ enumeration phase for all minimal vertex covers, combined with the well-known $\mathcal{O}^*(3^k)$ Dreyfus-Wagner algorithm [9], the running time of [19] follows.

Recently, the running time of the Dreyfus-Wagner algorithm has been improved to $\mathcal{O}^*((2 + \varepsilon)^k)$ for any $\varepsilon > 0$ (the smaller the ε , the bigger the polynomial, but this is hidden in the \mathcal{O}^* notation) [14, 30]. Combining this with the $\mathcal{O}^*(2^k)$ enumeration phase for all minimal vertex covers, we obtain an $\mathcal{O}^*((4 + \varepsilon)^k)$ algorithm for CVC-D.

This can be further improved by using a colouring scheme similar to the one given for 2-TVC-D in combination with *catalytic branching*, a technique introduced in [10]. This means that we consider all n cases as to whether a given vertex belongs to the vertex cover set to be constructed. In our case, we will mark a vertex selected in the branching. This can be seen in Alg. 3. Notice that the procedure CVC-annotated takes, besides the parameter(s), a non-empty set of marked vertices as arguments.

The procedure that deals with annotated instances is described in the following.

The reduction we use is described in Alg. 5. Let us briefly argue for the validity of the rules. Notice that all rules but the last work in the immediate neighbourhood of the catalyst vertex v .

1. Two marked neighboured vertices together are obviously covering all edges outgoing from either of them; this can be equivalently expressed by merging the two of them into a new marked vertex.
2. An unmarked neighbour x of degree 1 of a 1-marked vertex v has no need to go into the cover: the only edge it might cover is already covered by v and it cannot connect to other marked vertices. Hence, we can safely delete x .

Algorithm 3 An advanced search tree algorithm for CVC-D, called CVC-ST

Require: a graph $G = (V, E)$ and a non-negative integer k **Ensure:** return YES if G has a connected vertex cover of size at most k : NO otherwise

```
  if  $E = \emptyset$  then
    return YES;
  else
    for all  $v \in V$  do
5:     if CVC-annotated  $(G, k, k, \{v\}) = \text{YES}$  then
        return YES;
      end if
    end for
    return NO;
10: end if
```

Algorithm 4 A search tree algorithm for annotated CVC-D, called CVC-annotated

Require: a graph $G = (V, E)$, two integers k_1, k_2 (k_1 is bounding the first vertex-cover-like search tree part, while k_2 is bounding the number of terminal points in the subsequent Steiner tree algorithm; hence $k_1 \leq k_2$); a non-empty set of vertices C assumed to be marked**Ensure:** return YES if G has a connected vertex cover of size at most k that contains all vertices from C ; NO otherwise

```
  if  $k_1 < 0$  then
    return NO;
  else if  $E = \emptyset$  then
    return (Steiner-TREE  $(G, C) + k_2 \leq k_1$ );
5:  {Steiner-TREE computes a Steiner tree for  $G$  with (at most  $k_2$  many) terminal vertices  $C$ 
    and returns the number of Steiner points}
  else
    produce an irreducible instance; {for simplicity, we use the same namings as before: the
    reduction is described below}
    if  $k_1 < 0$  then
      return NO;
10:  else if there are no uncovered edges, i.e.,  $E = \emptyset$  then
      return (Steiner-TREE  $(G, C) + k_2 \leq k_1$ );
    end if
    pick a vertex  $u$  to branch at; {how to choose is described below in detail}
    if CVC-annotated  $(G, k_1 - 1, k_2, C \cup \{u\}) = \text{YES}$  then
15:     return YES;
    else
      return CVC-annotated  $(G - u, k_1 - \text{deg}(u), k_2, C \cup N(u))$ ;
    end if
  end if
```

3. An unmarked neighbour x of degree at least 2 of a marked vertex v that has an unmarked neighbour $y \neq v$ that is of degree at most 2 might be responsible for covering the edge $e = xy$. In principle, e could be also covered by y . However, if y was in the cover, then one of its at most two neighbours must go into the cover to satisfy the connectivity requirement. If the degree of y was 1, this means that x must be in the cover, which is just the case as claimed by the reduction rule. If y is of degree 2, then y could be in the cover together with another neighbour $z \neq x$ of y . Moreover, there must be a path from z to v within the cover that does not contain y . However, instead of putting v, y, z into the connected cover, we could also take v, x, z into an alternative connected cover of the same size.

Notice that after exhaustively applying the reduction rules, the marked vertices together form an independent set. Therefore, a reduced graph on which we start the Steiner tree phase is bipartite. However the bipartite property alone does not in general lead to a polynomial-time algorithm for this phase. To see this, it is straightforward to observe that Karp's reduction [24] shows that the Steiner tree problem is NP-hard even if we have a set T of terminal vertices such that G has no edges between two vertices in T and between two vertices in $V \setminus T$. This fact justifies the use of the $\mathcal{O}^*((2 + \varepsilon)^k)$ algorithm for the Steiner tree problem [14, 30] here.

Algorithm 5 Reductions for annotated CVC-D, called CVC-reduce

Require: a graph $G = (V, E)$, an integer k ; the set of marked vertices C

Ensure: return an irreducible instance

```

repeat
  if  $\exists x, v \in C : x \in N(v)$  then
    merge  $v$  and  $x$  into a new member of  $C$ ;
     $k_2 := k_2 - 1$ ; {the number of vertices to be considered in the Steiner tree phase of the
    overall algorithm is reduced}
5:  else if  $v \in C$  has an (unmarked) neighbour  $x$  of degree 1 then
    delete  $x$ ;
    else if  $v \in C$  has an unmarked neighbour  $x$  of degree at least 2 that has an unmarked
    neighbour  $y \neq v$  that is of degree at most 2 then
      mark  $x$ ;
       $k_1 := k_1 - 1$ ;
10:  end if
until no more changes occur to the instance

```

We now describe how to pick a vertex $u \notin C$ to branch at, i.e., either take u into the cover or all its neighbours. As before, let C be the cover found (assumed) so far (i.e., the set of marked vertices). If none of the listed conditions applies, we already found a vertex cover of G and can now run some Steiner tree algorithm to ensure connectivity of the cover set.

if possible then

Choose a vertex $v \in C$ such that there is a neighbour $u \in N(v)$ that has at least two unmarked neighbours

else if possible then

Choose a vertex $v \in C$ such that there is a vertex u in $N(N(v)) \setminus (N[v] \cup C)$ with at least one marked neighbour

else if possible then

Choose a vertex $v \in C$ such that there is a vertex u of degree at least 3 in $N(N(v)) \setminus (N[v] \cup C)$

end if

Why can we continue with the Steiner tree phase when none of the possibilities applies? Consider the vicinity of a vertex $v \in C$. By the reduction rules, all neighbours of v are unmarked. If v has a neighbour with at least two unmarked neighbours, we would branch. Hence, thereafter any neighbour x of v has at most one unmarked neighbour y . Conversely, by the reduction rules, x must have one neighbour apart from v . If all neighbours of x are marked, there is no need for further branching in the first phase, since all edges are covered, and if this observation applies to all neighbours of all $v \in V$, then the second, Steiner tree phase can start. Hence, assume that x has exactly one neighbour y that is unmarked. If y has a marked neighbour z , the second branching scenario applies unless $z = v$, a case already covered by the last reduction rule. If y has at most one neighbour besides x , the reduction rules would have triggered. Hence, y has at least two unmarked neighbours besides x , and the third branching scenario considers this final case.

As for the running time analysis, notice that there are now the following possible worst cases:

1. $v \in C$ has a neighbour u of degree ≥ 3 we branch at (with ≥ 2 unmarked neighbours).
2. $v \in C$ has a neighbour x of degree ≥ 2 that has an unmarked neighbour u with at least one marked neighbour z .
3. $v \in C$ has a neighbour of degree ≥ 2 that has an unmarked neighbour u with ≥ 3 unmarked vertices to branch at.

In Case 1, we can estimate the running time of the overall search tree (including the Steiner tree computation phase) as follows:

$$T(k_1, k_2) \leq T(k_1 - 1, k_2 - 1) + T(k_1 - 2, k_2).$$

The first term describes the running time of the branch that includes u into the cover and the second term describes the running time of the branch that excludes u from the cover but takes all neighbours into the cover. Notice that if u is taken into the cover, it will be marked and hence merged with its marked neighbour after the recursive call due to the reduction rules. Hence, the second parameter k_2 (bounding the Steiner tree part) is decremented as claimed. A little algebra shows that $T(k_1, k_2) \leq 1.2808^{k_1} (2 + \varepsilon)^{k_2}$.

In Case 2, when u is not put into the cover, x will go into the cover. In either case, the resulting marked vertex will be neighbour of a marked vertex, i.e., in the recursion the reduction rules trigger and reduce the second parameter. Hence, we can estimate

$$T(k_1, k_2) \leq 2T(k_1 - 1, k_2 - 1).$$

Due to the very nice reduction of the second parameter, we can estimate $T(k_1, k_2) \leq (2 + \varepsilon)^{k_2}$.

In Case 3, notice that in the case that u is not put into the cover but all its neighbours, a neighbour of v will be put into the cover which reduces the second parameter k_2 . $T(k_1, k_2) \leq T(k_1 - 1, k_2) + T(k_1 - 3, k_2 - 1) \leq 1.4655^{k_1} (2 + \varepsilon)^{k_2}$

This gives the claimed worst case running time. \square

5 Complexity and approximability of t -TEC

Let $G = (V, E)$ be a connected graph, where $n = |V|$, $m = |E| \geq 1$, and let $1 \leq t \leq n - 1$. We begin this section by presenting a Gallai identity involving the concepts of a t -tec and a t -tree packing. A t -tree packing of G is a collection $\mathcal{P} = \{G_1, \dots, G_k\}$ of vertex-disjoint (non-induced) subgraphs of G , each of which is a tree containing exactly t edges. The

value k is defined to be the *size* of \mathcal{P} . Let $\beta_{1,t}(G)$ denote the maximum size of a t -tree packing of G . Then $\beta_{1,1}(G) = \beta_1(G)$, the size of a maximum matching in G . The following result gives a Gallai identity involving $\alpha_{1,t}(G)$ and $\beta_{1,t}(G)$.

Theorem 11. *Let $G = (V, E)$ be a connected graph, where $n = |V|$, $m = |E| \geq 1$, and let $1 \leq t \leq n - 1$. Then $\alpha_{1,t}(G) + \beta_{1,t}(G) = n$.*

Proof. Let $\mathcal{P} = \{G_1, \dots, G_k\}$ be a t -tree packing of G such that $k = \beta_{1,t}(G)$. Let S initially contain the edges belonging to the subgraphs in \mathcal{P} . Then $|S| = kt$ and S covers $k(t + 1)$ vertices of G , so that $n - k(t + 1)$ vertices are as yet uncovered. Pick any uncovered vertex v . Then v is at distance at most t from a covered vertex w , for otherwise we contradict the maximality of \mathcal{P} . Let $v_0 = v$, and let v_0, v_1, \dots, v_s be the vertices (in order) on a path in G from v_0 to v_s , where v_s is covered, v_i is uncovered ($1 \leq i \leq s - 1$), and $1 \leq s \leq t$. Add $\{v_i, v_{i+1}\}$ to S ($0 \leq i \leq s - 1$). Continue in this way until all vertices are covered. Then S is a t -tec of G . Moreover we add one edge for every additional vertex that we cover, so that $|S| = kt + (n - k(t + 1)) = n - k$, i.e. $\alpha_{1,t}(G) \leq n - \beta_{1,t}(G)$.

Conversely let $\mathcal{S} = \{S : S \text{ is a } t\text{-tec in } G \text{ and } |S| = \alpha_{1,t}(G)\}$. Choose $S \in \mathcal{S}$ such that $G[S]$ contains the fewest number of cycles. Let $G_i = (V_i, S_i)$ ($1 \leq i \leq k$) be the connected components of $G[S]$, for some $k \geq 1$. Let i ($1 \leq i \leq k$) be given. Then by definition of S , it follows that G_i contains at least t edges. Now suppose that G_i contains a cycle, and let e be any edge on this cycle. If $k = 1$ then $S' = S \setminus \{e\}$ is a connected subgraph of G that spans V , and hence $|S'| \geq n - 1$, so that S' is a t -tec, contradicting the minimality of S . Hence $k \geq 2$. Since S is an edge cover, there exists an edge $e' = \{u, v\}$ in G such that u is covered by G_i and v is covered by some G_j ($1 \leq j \neq i \leq k$). Let $S' = (S \setminus \{e\}) \cup \{e'\}$. Then S' is a t -tec, $|S'| = |S|$ and S' has one fewer cycle than S , contradicting the choice of S . Hence G_i is acyclic. It follows that $|S_i| = |V_i| - 1$, so that

$$|S| = \sum_{i=1}^k |S_i| = \sum_{i=1}^k (|V_i| - 1) = n - k.$$

Let $\mathcal{P} = \{H_1, \dots, H_k\}$ be formed by “pruning” each G_i in order to form a tree H_i containing exactly t edges (this may be carried out by repeatedly deleting edges incident to vertices of degree 1 in G_i , until exactly t edges remain). Then \mathcal{P} is a t -tree packing of G , and $|\mathcal{P}| = k = n - \alpha_{1,t}(G)$, so that $\beta_{1,t}(G) \geq n - \alpha_{1,t}(G)$. \square

We remark that, in the case $t = 1$, Theorem 11 gives the familiar Gallai identity $\alpha_1(G) + \beta_1(G) = n$ [16].

For each $t \geq 1$, let t -TREE PACKING denote the problem of computing $\beta_{1,t}(G)$, given a connected graph $G = (V, E)$, where $n = |V| \geq t + 1$. Let t -TREE PACKING-D denote the decision version of t -TREE PACKING. Kirkpatrick and Hell [25] proved the following result concerning t -TREE PACKING-D.

Theorem 12 ([25]). *For each $t \geq 2$, t -TREE PACKING-D is \mathcal{NP} -complete.*

The following is an immediate consequence of Theorems 12 and 11.

Corollary 13. *For each $t \geq 2$, t -TEC-D is \mathcal{NP} -complete.*

The next two results concern the approximability of t -TEC for $t \geq 2$.

Theorem 14. *For each $t \geq 2$, t -TEC is approximable within 2.*

Proof. Let $G = (V, E)$ be an instance of t -TEC (a connected graph where $n = |V|$ and $m = |E| \geq t$). Any edge cover S of G satisfies $|S| \geq \frac{n}{2}$, since each edge of S covers 2 vertices of G . Now let T be a spanning tree of G . Suppose firstly that $t \leq n - 1$. Then T is a t -tec of G and $|T| = n - 1 \leq 2\alpha_1(G) \leq 2\alpha_{1,t}(G)$ by Proposition 2, as required. Now suppose that $t > n - 1$. Let $t' = t - (n - 1)$. As G is connected, we may construct a t -tec S by adding t' edges to T . Then $|S| = t$, so that S is in fact a minimum t -tec by Proposition 2. \square

Theorem 15. *2-TEC in bounded degree graphs is not approximable within some $\delta > 1$ unless $\mathcal{P} = \mathcal{NP}$.*

Proof. 2-TREE PACKING in graphs of maximum degree B is not approximable within some $\varepsilon > 1$ unless $\mathcal{P} = \mathcal{NP}$ [23]. We may consider this problem as the starting point for a reduction to 2-TEC that essentially follows the same lines as the proof of Theorem 11 in the case that $t = 2$ and $G = (V, E)$ is a connected graph of maximum degree B , where $n = |V|$ and $m = |E|$. Now $\beta_{1,2}(G) \geq m/(3B - 1)$, since any P_3 in a 2-tree packing \mathcal{P} of G rules out at most $3B - 1$ edges for inclusion in some other P_3 in \mathcal{P} . By Theorem 11, $\alpha_{1,2}(G) + \beta_{1,2}(G) = n \leq m$, since the graph constructed by Kann's reduction [23] is not acyclic. Hence the reduction described here is an L-reduction (see [33]) with parameters $\alpha = 3B - 2$ and $\beta = 1$. The result follows by [37, Theorem 63]. \square

We now consider the parameterized complexity of t -TEC ($t \geq 2$).

Theorem 16. *For each $t \geq 2$, t -TEC-D is in \mathcal{FPT} .*

Proof. Let $\langle G, k \rangle$ be an instance of t -TEC-D. Then k is a parameter and $G = (V, E)$ is a connected graph where $n = |V|$ and $m = |E| \geq t$. As observed in the proof of Theorem 14, $k \geq \frac{n}{2}$ or else $\langle G, k \rangle$ is a NO-instance. Hence $n \leq 2k$, so $m \leq (2k)^2$. Generating every subset S of E with at most k edges and verifying whether S is a t -tec is a process that takes $\mathcal{O}^*((2k)^{2k})$ overall time. \square

We now consider the concept of *parametric duality* (see [4, 11] for a recent exposition), which is in a sense quite related to the family of Gallai identities proved above. Define DUAL- t -TEC-D to be the problem of deciding, given a connected graph $G = (V, E)$ where $n = |V|$ and $m = |E| \geq t$, and a (dual) parameter k_d , whether there a t -tec of size at most $n - k_d$. Using the fact that 2-TREE-PACKING-D is in \mathcal{FPT} and solvable in time $\mathcal{O}^*(2^{5.3k})$, where k is the size of the 2-tree-packing [34], Theorem 11 implies the following result.

Theorem 17. *DUAL-2-TEC-D is in \mathcal{FPT} and can be solved in time $\mathcal{O}^*(2^{5.3k_d})$.*

Theorems 16 and 17 therefore imply that both 2-TEC-D and DUAL-2-TEC-D are in \mathcal{FPT} , a result rarely observed in the context of parameterized complexity. However, in the case of t -TVC and CVC, we can show:

Theorem 18. *DUAL-2-TVC-D is $\mathcal{W}[1]$ -complete. Also DUAL- t -TVC-D ($t \geq 3$) and DUAL-CVC-D are $\mathcal{W}[1]$ -hard.*

Proof. To show membership in $\mathcal{W}[1]$ of DUAL-2-TVC-D, we employ the ‘‘Turing way’’ [3, 11]. That is, we exhibit a Turing machine whose $f(k_d)$ -step halting problem is solvable if and only if the given instance of DUAL-2-TVC-D is a YES-instance.

A 1-tape nondeterministic Turing machine M_G for graph $G = (V, E)$ would work as follows. The tape alphabet is $V \times \{0, 1\}$ (plus the end markers).

1. Guess k_d letters from $V \times \{0\}$ and write them on the tape.

2. Sweep back and forth on the tape and verify that the vertices are independent. (If two vertices u, v have been guessed with $u \in N(v)$, then the Turing machine would enter an infinite loop.)

The second part of the tape alphabet can be used to protocol which two vertices are tested.

If all pairs have been tested, then the tape contains an independent set I .

3. Now use the second part of the tape alphabet to cycle through all subsets of I . For each subset $\emptyset \neq X \subseteq I$, we have to test whether $X = N(v)$ for some $v \notin I$. If this is the case, then we have detected a vertex from the vertex cover $V \setminus I$ that has no neighbour from $V \setminus I$.

To this end, an n -bit internal memory is used. Initially, this is an all-zero vector. Upon reading X off the tape, at most k_d bits are set to 1. Then, by the internal memory bit vector $X = N(v)$ can be checked in one further step. If the infinite loop is not entered (i.e., $X \neq N(v)$ for all $v \in V \setminus I$), then the k_d bits are set to 0 again, and then the “next set” is selected by the bit vector counter on the tape.

Finally, the bit vector counter on the tape contains only ones, and then the machine will stop.

Hence, there is a function $f(k_d)$ such that G has a total vertex cover of size $n - k_d$ iff M_G stops in at most $f(k_d)$ steps.

We now show that DUAL- t -TVC-D is $\mathcal{W}[1]$ -hard, for each $t \geq 2$. We use the same reduction as in Theorem 3, where $G = (V, E)$ is a connected graph with $n = |V| \geq 2$ and k_d is a parameter, given as an instance of INDEPENDENT SET-D. Then G has an independent set of size k_d if and only if the $(n + t)$ -vertex graph G' has a t -tvc of size $n - k_d + (t - 1) = (n + t) - (k_d + 1)$.

In the case of DUAL-CVC-D, the proof is similar; the same reduction may be used with $t = 2$. □

6 Concluding remarks

In this paper we have defined the concepts of a t -tvc and a t -tec for $t \geq 1$, which are motivated by the notions of covering and clustering in graphs. We have presented \mathcal{NP} -completeness, approximability and parameterized complexity results for associated optimization and decision problems.

Until now, enumeration-based solutions to parameterized decision problems seemed to be doomed to give rise to a complexity function $\mathcal{O}^*(C^k)$ where C is quite large. Our \mathcal{FPT} algorithms in this paper demonstrate how this can be overcome by introducing appropriate “colourings” and corresponding reduction rules within the search tree algorithm. A further example is the EDGE DOMINATING SET algorithm described in [12]. Moreover, a novel way of analyzing search trees that can be decomposed into two phases is exhibited; this has proved to be highly effective in the case of 2-TVC-D and CVC-D, and should also be applicable in improving the analysis of other fixed-parameter algorithms.

In Section 4, we presented an $\mathcal{O}^*(2.9316^k)$ algorithm for CVC-D. As mentioned in Section 1, an improved $\mathcal{O}^*(2.7606^k)$ algorithm for CVC-D will be reported in [29]. It is likely that a further improvement could be obtained by combining the approach of Moelle et al. with the reduction rules that we employ for our CVC-D algorithm.

The results in this paper leave open the following problems, among others, that are worthy of further consideration: (1) Formulate polynomial-time algorithms for t -TVC and

t -TEC in restricted classes of graphs. (2) Formulate (if possible) \mathcal{FPT} algorithms for t -TVC-D ($t > 2$). Are the corresponding parametric dual problems in $\mathcal{W}[1]$? (3) Consider “clustering” variants of vertex domination and edge domination.

Acknowledgements

The second author would like to thank Michele Zito for helpful discussions regarding 2-total vertex covers, and Pavol Hell for drawing our attention to reference [25] in connection with t -tree packings.

References

- [1] E.M. Arkin, M.M. Halldórsson, and R. Hassin. Approximating the tree and tour covers of a graph. *Information Processing Letters*, 47:275–282, 1993.
- [2] J.R.S. Blair. Personal communication, 2001.
- [3] M. Cesati. The Turing way to parameterized complexity. *Journal of Computer and System Sciences*, 67:654–685, 2003.
- [4] J. Chen, H. Fernau, I. A. Kanj, and G. Xia. Parametric duality and kernelization: Lower bounds and upper bounds on kernel size. In *Proceedings of STACS 2005: the 22nd Annual Symposium on Theoretical Aspects of Computer Science*, volume 3404 of *Lecture Notes in Computer Science*, pages 269–280. Springer, 2005.
- [5] M. Chlebík and J. Chlebíková. Complexity of approximating bounded variants of optimization problems. *Theoretical Computer Science*, 354(3):320–338, 2006.
- [6] E.J. Cockayne, R.M. Dawes, and S.T. Hedetniemi. Total domination in graphs. *Networks*, 10:211–219, 1980.
- [7] P. Damaschke. Parameterized enumeration, transversals, and imperfect phylogeny reconstruction. *Theoretical Computer Science*, 351:337–350, 2006.
- [8] I. Dinur and S. Safra. On the hardness of approximating minimum vertex cover. *Annals of Mathematics*, 162(1):439–485, 2005.
- [9] S. E. Dreyfus and R. A. Wagner. The Steiner problem in graphs. *Networks*, 1:195–207, 1972.
- [10] M. R. Fellows, C. McCartin, F. A. Rosamond, and U. Stege. Coordinatized kernels and catalytic reductions: an improved FPT algorithm for Max Leaf Spanning Tree and other problems. In *Proceedings of FST TCS 2000: the 20th Conference on Foundations of Software Technology and Theoretical Computer Science*, volume 1974 of *Lecture Notes in Computer Science*, pages 240–251. Springer, 2000.
- [11] H. Fernau. Parameterized algorithmics: A graph-theoretic approach. Habilitationsschrift, University of Tübingen, 2005.
- [12] H. Fernau. Edge dominating set: efficient enumeration-based exact algorithms. In *Proceedings of IWPEC 2006: the Third International Workshop on Parameterized and Exact Computation* (to appear), volume 4169 of *Lecture Notes in Computer Science*, pages 142–153. Springer, 2006.

- [13] F. Fomin, D. Kratsch, and G. Woeginger. Exact (exponential) algorithms for the dominating set problem. In *Proceedings of WG '04: the 30th International Workshop on Graph-Theoretic Concepts in Computer Science*, volume 3353 of *Lecture Notes in Computer Science*, pages 245–256. Springer, 2004.
- [14] B. Fuchs, W. Kern, Daniel Mölle, S. Richter, P. Rossmanith, and X. Wang. Dynamic programming for minimum Steiner trees. Technical Report zaik2005-492, University of Cologne, 2005.
- [15] M. Gaertler. Clustering. In U. Brandes and T. Erlebach, editors, *Network Analysis*, volume 3418 of *Lecture Notes in Computer Science*, chapter 8, pages 178–215. Springer, 2005.
- [16] T. Gallai. Über extreme Punkt-und Kantenmengen. *Ann. Univ. Sci. Budapest, Eötvös Sect. Math.*, 2:133–138, 1959.
- [17] M.R. Garey and D.S. Johnson. The rectilinear Steiner tree problem is NP-complete. *SIAM Journal on Applied Mathematics*, 32(4):826–834, 1977.
- [18] M.R. Garey, D.S. Johnson, and L. Stockmeyer. Some simplified NP-complete graph problems. *Theoretical Computer Science*, 1:237–267, 1976.
- [19] J. Guo, R. Niedermeier, and S. Wernicke. Parameterized complexity of generalized vertex cover problems. In *Proceedings of WADS 2005: the 9th International Workshop on Algorithms and Data Structures*, volume 3608 of *Lecture Notes in Computer Science*, pages 36–48. Springer, 2005.
- [20] F. Harary. *Graph Theory*. Addison-Wesley, 1969.
- [21] T. Haynes, S.T. Hedetniemi, and P.J. Slater, editors. *Domination in Graphs: Advanced Topics*. Marcel Dekker, 1998.
- [22] T. Haynes, S.T. Hedetniemi, and P.J. Slater. *Fundamentals of Domination in Graphs*. Marcel Dekker, 1998.
- [23] V. Kann. Maximum bounded H-matching is MAX SNP-complete. *Information Processing Letters*, 49:309–318, 1994.
- [24] R.M. Karp. Reducibility among combinatorial problems. In R.E. Miller and J.W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.
- [25] D.G. Kirkpatrick and P. Hell. On the completeness of a generalized matching problem. In *Proceedings of STOC '78: the 10th Annual ACM Symposium on Theory of Computing*, pages 240–245. ACM, 1978.
- [26] D.F. Manlove. On the algorithmic complexity of twelve covering and independence parameters of graphs. *Discrete Applied Mathematics*, 91:155–175, 1999.
- [27] S. Mitchell and S.T. Hedetniemi. Edge domination in trees. In *Proceedings of the 8th South-Eastern Conference on Combinatorics, Graph Theory and Computing*, pages 489–509. Utilitas Mathematica, 1977.
- [28] D. Mölle, S. Richter, and P. Rossmanith. Enumerate and expand: Improved algorithms for connected vertex cover and tree cover. In *Proceedings of CSR '06: International Computer Science Symposium in Russia*, volume 3967 of *Lecture Notes in Computer Science*, pages 280–290. Springer, 2006.

- [29] D. Mölle, S. Richter, and P. Rossmanith. Enumerate and expand: New runtime bounds for vertex cover variants. In *Proceedings of COCOON 2006: the 12th Annual International Computing and Combinatorics Conference* (to appear), *Lecture Notes in Computer Science*. Springer, 2006.
- [30] D. Mölle, S. Richter, and P. Rossmanith. A faster algorithm for the Steiner Tree problem. In *Proceedings of STACS 2006: the 23rd Annual Symposium on Theoretical Aspects of Computer Science*, volume 3884 of *Lecture Notes in Computer Science*, pages 561–570. Springer, 2006.
- [31] N. Nishimura, P. Ragde, and D.M. Thilikos. Fast fixed-parameter tractable algorithms for nontrivial generalizations of vertex cover. *Discrete Applied Mathematics*, 152:229–245, 2005.
- [32] R.Z. Norman and M.O. Rabin. An algorithm for a minimum cover of a graph. *Proceedings of the American Mathematical Society*, 10:315–319, 1959.
- [33] C.H. Papadimitriou and M. Yannakakis. Optimization, approximation and complexity classes. *Journal of Computer and System Sciences*, 43(3):425–440, 1991.
- [34] E. Prieto and C. Sloper. Looking at the stars. *Theoretical Computer Science*, 351:437–445, 2006.
- [35] C. Savage. Depth-first search and the vertex cover problem. *Information Processing Letters*, 14:233–235, 1982.
- [36] S. Ueno, Y. Kajitani, and S. Gotoh. On the nonseparating independent set problem and feedback set problem for graphs with no vertex degree exceeding three. *Discrete Mathematics*, 72:355–360, 1988.
- [37] M. Zito. *Randomised Techniques in Combinatorial Algorithms*. PhD thesis, University of Warwick, Department of Computer Science, 1999.