Vector Connectivity in Graphs*

Endre Boros

MSIS Department and RUTCOR, Rutgers University,

100 Rockafeller Road, Piscataway, NJ 08854

{boros}@rutcor.rutgers.edu

Pinar Heggernes, Pim van 't Hof

Department of Informatics, University of Bergen, Norway

{pinar.heggernes,pim.vanthof}@ii.uib.no

Martin Milanič

University of Primorska, UP IAM, Muzejski trg 2, SI6000 Koper, Slovenia

University of Primorska, UP FAMNIT, Glagoljaška 8, SI6000 Koper, Slovenia

martin.milanic@upr.si

December 16, 2013

Abstract

Motivated by challenges related to domination, connectivity, and information propagation in social and other networks, we initiate the study of the Vector Connectivity problem. This problem takes as input a graph G and an integer k_v for every vertex v of G, and the objective is to find a vertex subset S of minimum cardinality such that every vertex v either belongs to S, or is connected to at least k_v vertices of S by disjoint paths. If we require each path to be of length exactly 1, we

*An extended abstract of this paper appeared in the proceedings of the 10th Annual Conference on Theory and Applications of Models of Computation (TAMC 2013) [1].

get the well-known Vector Domination problem, which is a generalization of the famous Dominating Set problem and several of its variants. Consequently, our problem becomes NP-hard if an upper bound on the length of the disjoint paths is also supplied as input. Due to the hardness of these domination variants even on restricted graph classes, like split graphs, Vector Connectivity seems to be a natural problem to study for drawing the boundaries of tractability for this type of problems. We show that Vector Connectivity can actually be solved in polynomial time on split graphs, in addition to cographs and trees. We also show that the problem can be approximated in polynomial time within a factor of $\ln n + 2$ on all n-vertex graphs.

Keywords: vector connectivity, approximation algorithm, polynomial time algorithm, split graph, cograph, tree

1 Introduction and Motivation

Connectivity between parts of a graph via disjoint paths is one of the best studied subjects in graph theory and graph algorithms, where Network Flow and Disjoint Paths and many of their variants are among the most well-known problems. In this paper, we introduce, motivate, and study a natural network problem, which we call Vector Connectivity. Given a graph $G = (V, E)$ and a vector $\boldsymbol{k}$ indexed by the vertices of G, such that $\boldsymbol{k} = (k_v : v \in V)$ and k_v is between 0 and the degree of v for each vertex $v \in V$, the task of Vector Connectivity is to find a set $S \subseteq V$ of minimum cardinality that satisfies the following: every vertex v of G is either in S or is connected to at least k_v vertices of S via paths that pairwise intersect in no other vertex than v.

In Vector Connectivity there is no restriction on the lengths of the involved disjoint paths. If each path is restricted to be of length exactly 1, we get the well-known Vector Domination problem; this problem was introduced by Harant et al. [11] as a generalization of the classical problems Dominating Set and Vertex Cover. The Dominating Set problem and its variants

have been studied extensively, as they naturally appear in a wide variety of theoretical and practical applications. This has led to a vast amount of papers and several books on domination, e.g., [12, 13]. Dominating Set and hence Vector Domination are also among the toughest NP-hard problems as they remain NP-hard on various classes of graphs, such as planar graphs of maximum degree 3, bipartite graphs, and most interesting for our study: split graphs [7, 13]. The popularity and the difficulty of these domination problems, the connection between Vector Domination and Vector Connectivity, and the question whether allowing paths of unbounded length rather than direct edges or bounded-length paths can result in tractability, are among the motivations for studying the Vector Connectivity problem.

Chlebík and Chlebíková [3] showed that Dominating Set, and consequently Vector Domination, cannot be approximated in polynomial time within a factor of $(1-\epsilon)\ln n$ for any constant $\epsilon > 0$ on n-vertex graphs unless NP $\subseteq$ DTIME($n^{O(\log\log n)}$), even when restricted to the class of bipartite graphs or split graphs. On the positive side, Cicalese et al. [4] presented a greedy algorithm for Vector Domination with approximation factor $\ln(2\Delta)+1$, where Δ denotes the maximum degree of the input graph. Moreover, they showed that the problem can be solved in polynomial time on trees and cographs. If one asks for disjoint paths of bounded length rather than direct edges, it is not known in general whether the problem can be approximated within a factor of $O(\log n)$. This gives another motivation to study the unbounded-length paths case, which is exactly the Vector Connectivity problem.

In this paper, we show that Vector Connectivity can be approximated within a factor of $\ln n + 2$ in polynomial time on general graphs, which we find interesting due to the known and unknown approximation results mentioned above. Furthermore, we show that Vector Connectivity can be solved in polynomial time on split graphs, cographs, and trees. We find in particular the tractability result on split graphs surprising, as it is in contrast with the aforementioned NP-hardness and inapproximability results for the Dominating Set problem on split graphs. Furthermore, these intractability results imply

that if paths are required to be of length at most an input bound then the problem remains NP-hard on split graphs. However, split graphs do not have any induced paths of length 4 or more. Hence our positive result on split graphs implies that the bounded-length path version of Vector Connectivity, which is a generalization of Vector Domination, is solvable in polynomial time on split graphs if the bound is at least 3.

Note that the classes of split graphs, cographs, and trees are all subclasses of perfect graphs, but they are not contained in each other. They form some of the most studied graph classes on which many algorithms have been given, and they play the main role in several books, e.g., in the monograph on perfect graphs by Golumbic [8], and in the monograph by Mahadev and Peled [15] on threshold graphs, which form a subclass of both split graphs and cographs.

Before we proceed to the technical part presenting and proving our results, we end this section by mentioning another motivation, which comes from information propagation in social networks. One famous problem of this type is Target Set Selection (see, e.g., [2, 14, 16]), where every vertex v has a threshold t_v such that v gets activated if at least t_v of its neighbors are activated, and the task is to select a minimum cardinality vertex subset that results in the activation of all vertices eventually. The practical application behind this problem is the desire by manufacturers to give away their products to a selected small group of people, based on the scenario that every potential customer will decide to buy the product if he or she has enough friends who possess the product. Another possible scenario can be that every potential customer will decide to buy the product only if he or she has enough independent ways to learn about the product. Vector Connectivity fits into this scenario if we assume that information spreads freely along the paths of the network.

2 Definitions and Notation

Unless otherwise stated, we work with undirected simple graphs $G = (V, E)$, where V is the set of vertices, E is the set of edges, and $|V|$ is denoted by n.

We use standard graph terminology. In particular, the degree of a vertex v in G is denoted by $d_G(v)$, the maximum degree of a vertex in G is denoted by $\Delta(G)$, and $V(G)$ refers to the vertex set of G. For a given rooted tree T, we write T_v to denote the subtree rooted at vertex v, including vertex v. An *induced path* in a graph G is an induced subgraph of G isomorphic to a path.

Given a graph $G = (V, E)$, a set $S \subseteq V$ and a vertex $v \in V \setminus S$, a *v–S fan of order k* is a collection of k paths $P_1, \ldots, P_k$ such that (1) every P_i is a path connecting v to a vertex of S, and (2) the paths are pairwise vertex-disjoint except at v, i.e., $V(P_i) \cap V(P_j) = \{v\}$ holds for all $1 \leq i < j \leq k$. Given an integer-valued vector $\boldsymbol{k} = (k_v : v \in V)$ with $k_v \in \{0, 1, \ldots, d_G(v)\}$ for every $v \in V$, a *vector connectivity set* for $(G, \boldsymbol{k})$ is a set $S \subseteq V$ such that there exists a v–S fan of order k_v for every $v \in V \setminus S$. We say that k_v is the *requirement* of vertex v. The minimum size of a vector connectivity set for $(G, \boldsymbol{k})$ is denoted by $\kappa(G, \boldsymbol{k})$.

The Vector Connectivity problem is the problem of finding a vector connectivity set of minimum size, and can be formally stated as follows:

Vector Connectivity	
Input:	A graph $G = (V, E)$ and a vector $\boldsymbol{k} = (k_v : v \in V) \in \mathbb{Z}_+^V$ with $k_v \in \{0, 1, \ldots, d_G(v)\}$ for all $v \in V$.
Task:	Find a vector connectivity set for $(G, \boldsymbol{k})$ of size $\kappa(G, \boldsymbol{k})$.

For every $v \in V$ and every set $S \subseteq V \setminus \{v\}$, we say that v is *k-connected* to S if there is a v–S fan of order k in G. Hence, given an instance $(G, \boldsymbol{k})$ of Vector Connectivity, a set $S \subseteq V$ is a vector connectivity set for $(G, \boldsymbol{k})$ if and only if every $v \in V \setminus S$ is k_v-connected to S. For a subset $A \subseteq V$, we write $\boldsymbol{k}|_A$ to denote the sub-vector of $\boldsymbol{k}$ indexed by elements of A, and $\mathbf{1}_A$ denotes the all-one vector indexed by elements of A. We let $\sigma(v, A)$ to denote the maximum order of a v–A fan in G. In other words, $\sigma(v, A) = \max\{s \mid v \text{ is } s\text{-connected to } A\}$. Let $B \subseteq A$, let $\mathcal{A}$ be a v–A fan and let $\mathcal{B}$ be a v–B fan. We say that $\mathcal{A}$ *contains* $\mathcal{B}$ if the collection of paths in $\mathcal{B}$ is a subcollection of the paths in $\mathcal{A}$.

A set of vertices in a graph is a *clique* if they are all pairwise adjacent, and it is an *independent set* if no two of them are adjacent. A graph is a *split graph* if its vertex set can be partitioned into a clique C and an independent set I, where (C, I) is called a *split partition* of G. Split graphs can be recognized and a split partition can be computed in linear time [10].

For two vertex-disjoint graphs G_1 and G_2, $G_1 \oplus G_2$ denotes the *disjoint union* of G_1 and G_2, i.e., $G_1 \oplus G_2 = (V(G_1) \cup V(G_2), E(G_1) \cup E(G_2))$, and $G_1 \otimes G_2$ denotes the *join* of G_1 and G_2, i.e., the graph obtained by adding to $G_1 \oplus G_2$ all edges of the form $\{uv \mid u \in V(G_1), v \in V(G_2)\}$. The class of *cographs* is defined recursively through the following operations: a single vertex is a cograph; if G_1 and G_2 are vertex-disjoint cographs, then $G_1 \oplus G_2$ is a cograph; if G_1 and G_2 are vertex-disjoint cographs, then $G_1 \otimes G_2$ is a cograph. Cographs, split graphs, and trees are not related to each other inclusion-wise.

A well-known characterization of cographs is via cotrees. A *cotree* T of a cograph G is a rooted tree with two types of interior nodes, $\oplus$-nodes and $\otimes$-nodes, that has the following property: there is a bijection between the vertices of G and the leaves of T such that two vertices u and v are adjacent in G if and only if the lowest common ancestor of the leaves u and v in T is a $\otimes$-node. In particular, every node t of T corresponds to an induced subgraph of G, which is the disjoint union or the join of the subgraphs of G corresponding to the children of t. A graph is a cograph if and only if it has a cotree [5]. Cographs can be recognized and a cotree can be generated in linear time [6, 9]. For our purposes, it is convenient to use the binary version of a cotree, which is commonly used for algorithms on cographs: the recursive definition of cographs implies that we can assume the cotree to be binary. We will call this a *nice* cotree. Clearly, given a cotree of a cograph, a nice cotree can be obtained in linear time.

3 A Polynomial-Time Approximation Algorithm

In this section, we show that Vector Connectivity can be approximated in polynomial time by a factor of $\ln n + 2$ on all graphs. We will achieve this by

showing that Vector Connectivity can be recast as a particular case of the well-known Minimum Submodular Cover problem, which will allow us to apply a classical approximation result due to Wolsey [22].

First, we recall some definitions and results about submodular functions, hypergraphs and matroids that we will use in our proofs (see, e.g., [20]). Given a finite set U, a function $g : 2^U \to \mathbb{Z}_+$ is *submodular* if for every $X, Y \subseteq U$ with $X \subseteq Y$ and every $x \in U \setminus Y$, we have $g(Y \cup \{x\}) - g(Y) \leq g(X \cup \{x\}) - g(X)$. An instance of the (unweighted) Minimum Submodular Cover problem consists of a set U and an integer-valued, non-decreasing, submodular function $g : 2^U \to \mathbb{Z}_+$. The objective is to pick a set $S \subseteq U$ of minimum cardinality such that $g(S) = g(U)$.

A *hypergraph* is a pair $H = (U, \mathcal{E})$ where U is a finite set of *vertices* and $\mathcal{E}$ is a set of subsets of U, called *hyperedges*. A *matroid* is a hypergraph $M = (U, \mathcal{F})$ such that $\mathcal{F}$ is nonempty and closed under taking subsets, and its elements, called *independent sets*, satisfy the following "exchange property": for every two independent sets A and B such that $|A| < |B|$, there exists an element of B whose addition to A results in a larger independent set. (This is just one of the many equivalent ways to define matroids, see, e.g., [17, 21].) Given a matroid $M = (U, \mathcal{F})$, the *rank function* of M is the function $r_M : 2^U \to \mathbb{Z}_+$ that assigns to every subset S of U the maximum size of an independent set contained in S. The following property of rank functions of matroids is well known (see, e.g., [20]).

Lemma 1. *For every matroid M, its rank function r_M is submodular.*

A *gammoid* is a hypergraph $\Gamma = (U, \mathcal{E})$ derived from a triple (D, S, T) where $D = (V, A)$ is a digraph and $S, T \subseteq V$, such that $U = S$ and a subset S' of S forms a hyperedge if and only if there exist $|S'|$ vertex-disjoint directed paths in D connecting S' to a subset of T.

Lemma 2 ([18, 19])**.** *Every gammoid is a matroid.*

For any instance $(G = (V, E), \boldsymbol{k})$ of Vector Connectivity, we define a

function $f : 2^V \longrightarrow \mathbb{Z}_+$ as follows:

$$f(X) = \textstyle\sum_{v \in V} f_v(X)\,, \text{ where } X \subseteq V, \text{ and}$$

$$f_v(X) = \begin{cases} \min\{\sigma(v, X), k_v\} & \text{if } v \notin X; \\ k_v & \text{if } v \in X. \end{cases} \tag{1}$$

Observe that a set $S \subseteq V$ satisfies $f(S) = f(V)$ if and only if S is a vector connectivity set for $(G, \boldsymbol{k})$. Consequently, Lemma 3 below immediately implies that Vector Connectivity is a special case of Minimum Submodular Cover.

Lemma 3. *Let $(G = (V, E), \boldsymbol{k})$ be an instance of* Vector Connectivity. *Then the function $f : 2^V \longrightarrow \mathbb{Z}_+$, given by (1), satisfies the following properties:*

(i) $f(\emptyset) = 0$*;*

(ii) f *is integer-valued, i.e.,* $f(X) \in \mathbb{Z}_+$ *for every* $X \subseteq V$*;*

(iii) f *is non-decreasing, i.e.,* $f(X) \leq f(Y)$ *whenever* $X \subseteq Y \subseteq V$*;*

(iv) f *is submodular.*

Proof. It is easy to verify that properties (i)–(iii) hold. In order to show that f is submodular, it suffices to show that the function $f_v(\cdot)$ is submodular for every $v \in V$. Let $v \in V$, and let $C := \max\{\Delta(G), \max_{v \in V(G)}\{k_v\}\} + 1$. We define a function $g_v : 2^V \longrightarrow \mathbb{Z}_+$ as follows:

$$g_v(X) = \begin{cases} \sigma(v, X) & \text{if } v \notin X; \\ C & \text{if } v \in X. \end{cases}$$

It is easy to verify that g_v is non-decreasing. Moreover, we have $f_v(X) = \min\{g_v(X), k_v\}$ for every $X \subseteq V$. Therefore, in order to prove the submodularity of f_v, it suffices to prove that g_v is submodular (see, e.g., [20, p. 781]), which is equivalent to proving that for all $X \subseteq Y \subseteq V$ and for all $w \in V \setminus Y$,

$$g_v(Y \cup \{w\}) - g_v(Y) \leq g_v(X \cup \{w\}) - g_v(X)\,. \tag{2}$$

If $v \in Y$, then $g_v(Y) = g_v(Y \cup \{w\}) = C$, and the left-hand side of inequality (2) is equal to 0. Hence inequality (2) holds since g_v is non-decreasing. Similarly, if $w = v$, then $g_v(Y \cup \{w\}) = g_v(X \cup \{w\}) = C$, and inequality (2) holds since g_v is non-decreasing.

Now suppose that $w \notin Y \cup \{v\}$. Since $X \subseteq Y$, we also have that $w \notin X \cup \{v\}$. In this case, inequality (2) simplifies to

$$\sigma(v, Y \cup \{w\}) - \sigma(v, Y) \leq \sigma(v, X \cup \{w\}) - \sigma(v, X)\,. \tag{3}$$

In order to show that inequality (3) holds, it suffices to prove that the function $h_v : 2^{V \setminus \{v\}} \longrightarrow \mathbb{Z}_+$, defined by $h_v(W) = \sigma(v, W)$ for all $W \subseteq V \setminus \{v\}$, is submodular. Consider the gammoid Γ derived from the triple $(D, V \setminus \{v\}, N_G(v))$ where D is the digraph obtained from G by replacing each edge with a pair of oppositely directed arcs. By Lemma 2, Γ is a matroid. It follows directly from the definition that function h_v is equal to the rank function r_Γ of Γ. Therefore, by Lemma 1, the function h_v is submodular, which completes the proof of Lemma 3. □

Theorem 1. Vector Connectivity *can be approximated within a factor of* $\ln n + 2$ *in polynomial time.*

Proof. Let $(G = (V, E), \boldsymbol{k})$ be an instance of Vector Connectivity with $|V| = n$. From the definition of the function f, given by (1), it follows that a set $S \subseteq V$ satisfies $f(S) = f(V)$ if and only if S is a vector connectivity set for $(G, \boldsymbol{k})$. Hence, an optimal solution to the Vector Connectivity problem is provided by a minimum size subset $S \subseteq V$ such that $f(S) = f(V)$, i.e., by an optimal solution for Minimum Submodular Cover. An approximation to such a set S can be found in the following way.

Let $\mathbb{A}$ denote the natural greedy strategy which starts with $S = \emptyset$ and iteratively adds to S the element $v \in V \setminus S$ such that $f(S \cup \{v\}) - f(S)$ is maximum, until $f(S) = f(V)$ is achieved. The maximum order of a v–S fan can be computed in polynomial time using an easy reduction to the well-known Maximum Flow problem, and thus the function f is polynomially computable. There-

fore, the greedy strategy can be implemented in polynomial time. Moreover, Wolsey [22] proved that if f satisfies the four properties listed in Lemma 3, then algorithm $\mathbb{A}$ is an $H(\tau)$-approximation algorithm for Minimum Submodular Cover, and consequently for Vector Connectivity, where $H(j) = \sum_{i=1}^{j} \frac{1}{i}$ denotes the j-th harmonic number, and $\tau = \max_{y \in V} f(\{y\}) - f(\emptyset)$. For every $y \in V$, we have

$$f(\{y\}) = \sum_{v \in V \setminus \{y\}} f_v(\{y\}) + f_y(\{y\}) \leq n - 1 + k_y \leq n + \Delta(G)\,.$$

Since $f(\emptyset) = 0$, this implies $\tau \leq n + \Delta(G)$. Hence, algorithm $\mathbb{A}$ is an $H(n + \Delta(G))$-approximation algorithm for Vector Connectivity. Since $H(n) \leq \ln n + 1$ for $n \geq 1$, we can further bound the approximation ratio ρ of $\mathbb{A}$ from above as follows:

$$\rho \leq H(n + \Delta(G)) \leq \ln(n + \Delta(G)) + 1 \leq \ln(2n) + 1 = \ln n + \ln 2 + 1 \leq \ln n + 2\,,$$

yielding the desired result. □

4 A Polynomial-Time Algorithm for Split Graphs

Recall that the Vector Domination problem on split graphs is both NP-hard and hard to approximate within a factor of $(1 - \epsilon) \ln n$ for any constant $\epsilon > 0$. In this section, we give a polynomial-time algorithm to solve the Vector Connectivity problem on split graphs. Our algorithm is based on the following lemma.

Lemma 4. *Let $(G, \boldsymbol{k})$ be an instance of* Vector Connectivity*, where G is a split graph. Let S be any set of vertices in G such that $k_u \geq k_v$ for every pair of vertices $u \in S$ and $v \in V(G) \setminus S$. Then there exists a v–S fan of order $\min\{k_v, |S|\}$ for every $v \in V(G) \setminus S$.*

Proof. Let (C, I) be a split partition of $G = (V, E)$, and for convenience let $S_I = S \cap I$ and $S_C = S \cap C$. We will call the vertices of $V \setminus S$ *free vertices*. Let v be a free vertex of G. We first show that every vertex $u \in S_I$ has at least

$k_v - |S_C|$ free neighbors. To see this, let $u \in S_I$. It is obvious that u has at least $d_G(u) - |S_C|$ free neighbors. Since $u \in S$ and $v \in V \setminus S$, we have $k_v \leq k_u$. This, together with the assumption that $k_v \leq d_G(v)$ for every $v \in V$, implies that u has at least $d_G(u) - |S_C| \geq k_u - |S_C| \geq k_v - |S_C|$ free neighbors.

Suppose that v is a vertex of C. Every vertex of S_C is a neighbor of v, and thus v is $\min\{k_v, |S_C|\}$-connected to S_C. If $k_v \leq |S_C|$ then the lemma follows, so assume that $k_v > |S_C|$. Recall that every vertex $u \in S_I$ has at least $k_v - |S_C|$ free vertices in its neighborhood. Let $S' \subseteq S_I$ be any subset of S_I such that $|S'| = \min\{k_v - |S_C|, |S_I|\}$. Let G' be the bipartite subgraph of G obtained from the subgraph of G induced by $S' \cup (N_G(S') \setminus S_C)$ by deleting all edges of the form $\{xy \mid x, y \in N_G(S')\}$. Since $S' \subseteq S_I$, every vertex in S' has at least $k_v - |S_C|$ free neighbors in G'. Consequently, every non-empty subset $S'' \subseteq S'$ has at least $k_v - |S_C| \geq |S'| \geq |S''|$ neighbors in G', so Hall's Theorem implies that there is a matching M in G' that saturates S'. Let Y be the set of endpoints of M that are not in S'. Then $Y \subseteq C$, and it is possible that $v \in Y$. Since both v and all the vertices of Y belong to the clique C, v can reach at least $|S'| = \min\{k_v - |S_C|, |S_I|\}$ vertices of S_I via disjoint paths that do not contain vertices of S_C, using the edges of M. Consequently, v is $\min\{k_v, |S|\}$-connected to S, and the lemma follows.

Suppose now that v is a vertex of I. Since $k_v \leq d_G(v)$, v is $\min\{k_v, |S_C|\}$-connected to S_C. Let $\mathcal{P}_C$ be a v–S_C fan of order $\min\{k_v, |S_C|\}$ that is of smallest total path length. In particular, every path in $\mathcal{P}_C$ is of length 1 or 2. If $k_v \leq |S_C|$ then the lemma follows, so assume that $k_v > |S_C|$. In this case, exactly $|S_C|$ neighbors of v are used by the paths in $\mathcal{P}_C$. However, v has at least $d_G(v) - |S_C| \geq k_v - |S_C|$ additional neighbors that are free vertices in C. Furthermore, we already proved that every $u \in S_I$ has at least $k_v - |S_C|$ free vertices in its neighborhood. Each such vertex is either a neighbor of v or a neighbor of a neighbor of v. Thus v can reach at least $\min\{k_v - |S_C|, |S_I|\}$ vertices of S_I via disjoint paths that intersect each other and the paths of $\mathcal{P}_C$ only in vertex v. This shows that v is $\min\{k_v, |S|\}$-connected to S, and the lemma follows. □

Lemma 4 implies that we can sort the vertices of G by their $\boldsymbol{k}$-values in non-increasing order, and greedily pick vertices from the start of the sorted list to be in S until we have a vector connectivity set. This is formalized in the proof of the following theorem.

Theorem 2. Vector Connectivity *can be solved in polynomial time on split graphs.*

Proof. Let $(G, \boldsymbol{k})$ be an instance of Vector Connectivity, where $G = (V, E)$ is a split graph with split partition (C, I). Given a subset $S \subseteq V$ and a vertex $v \in V \setminus S$, we define $n(S, v) = k_v - \max\{t \mid v \text{ is } t\text{-connected to } S\}$. Then, $n(S, v) \leq 0$ if and only if v is k_v-connected to S. In particular, if $n(S, v) > 0$, then there exists a v–S fan of order $k_v - n(S, v)$ in G but no v–S fan of order at least $k_v - n(S, v) + 1$. Clearly, S is a vector connectivity set for $(G, \boldsymbol{k})$ if and only if $n(S, v) \leq 0$ for every $v \in V \setminus S$.

Lemma 4 suggests the following algorithm. We start by sorting the vertices of G as $v_1, v_2, \ldots, v_n$ so that $k_{v_i} \geq k_{v_j}$ for every $1 \leq i < j \leq n$. We set $S = \emptyset$ and $i = 0$. Then, as long as there are vertices $v \in V \setminus S$ such that $n(S, v) > 0$, we increase i by 1 and add vertex v_i to S. If $n(S, v) \leq 0$ for every vertex $v \in V \setminus S$, then the algorithm outputs the set S and terminates.

Let us prove that the algorithm is correct. Since the algorithm stops only when $n(S, v) \leq 0$ for every vertex $v \in V \setminus S$, the set S output by the algorithm is a vector connectivity set for $(G, \boldsymbol{k})$. It remains to show that there is no vector connectivity set S' for $(G, \boldsymbol{k})$ such that $|S'| < |S|$. As a result of Lemma 4, every time we add a vertex u to S, $n(S, v)$ does not increase for any vertex $v \in V \setminus S$, and $n(S, v)$ decreases by exactly 1 for every vertex $v \in V \setminus S$ for which $n(S, v) > 0$ before u was added to S. This implies that when the algorithm terminates, there is a vertex $v \in V \setminus S$ such that $k_v = |S|$. Consequently, any vector connectivity set for $(G, \boldsymbol{k})$ must either contain at least $|S|$ vertices or contain v. Let S' be a vector connectivity set for $(G, \boldsymbol{k})$ and assume, for contradiction, that $|S'| < |S|$. Then S' contains v by the above arguments, and there exists a vertex $u \in S \setminus S'$. Since $k_u \geq k_v = |S|$ and u is k_u-connected to S', we find that S' must contain

at least $|S|$ vertices, contradicting the assumption that $|S'| < |S|$. This finishes the correctness proof of the algorithm. It is clear that the algorithm runs in polynomial time. □

5 A Polynomial-Time Algorithm for Cographs

In this section we show that Vector Connectivity can be solved in polynomial time on cographs. In order to solve the original problem, we use a dynamic programming approach along a nice cotree representing the input cograph. (Recall that the notion of a nice cotree was defined on p. 6.) In fact, we need to solve the following more general variant of Vector Connectivity. For a graph $G = (V, E)$, an integer-valued vector $\boldsymbol{k} = (k_v : v \in V)$, and an integer ℓ, we say that a set $S \subseteq V$ is a *vector connectivity set for* $(G, \boldsymbol{k}, \ell)$ if S is a vector connectivity set for $(G, \boldsymbol{k})$ such that $v \in S$ whenever $k_v \geq \ell$. Let us denote by $\kappa(G, \boldsymbol{k}, \ell)$ the minimum size of a vector connectivity set for $(G, \boldsymbol{k}, \ell)$. Since $S = V$ is a vector connectivity set for $(G, \boldsymbol{k}, \ell)$, the above parameter is well defined and satisfies $\kappa(G, \boldsymbol{k}, \ell) \leq |V|$. Clearly, the following relation holds, and hence solving the described variant indeed also solves Vector Connectivity.

Lemma 5. $\kappa(G, \boldsymbol{k}) = \kappa(G, \boldsymbol{k}, \max_{v \in V} k_v + 1)$.

In order to simplify the presentation of our algorithm, we assume in this section that in the input to the Vector Connectivity problem and its variant mentioned above, requirements k_v are allowed to be negative. If $k_v < 0$, no condition is imposed on vertex v, and it can be treated the same as if $k_v = 0$.

The first lemma below is an easy observation.

Lemma 6. *Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be two graphs such that $V_1 \cap V_2 = \emptyset$, and let $G = G_1 \oplus G_2$. Then*

$$\kappa(G, \boldsymbol{k}, \ell) = \kappa(G_1, \boldsymbol{k}|_{V(G_1)}, \ell) + \kappa(G_2, \boldsymbol{k}|_{V(G_2)}, \ell)\,.$$

Lemma 7. *Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be two graphs such that $V_1 \cap V_2 = \emptyset$, and let $G = G_1 \otimes G_2$. Let $n_1 = |V_1|$ and $n_2 = |V_2|$, and let*

$\mathcal{F} = \{0, 1, \ldots, n_1\} \times \{0, 1, \ldots, n_2\}$. *Then, for every integer* ℓ*, we have*

$$\kappa(G, \boldsymbol{k}, \ell) = \min_{(i,j)\in\mathcal{F}} f(i, j)$$

with

$$f(i, j) = \max\left\{\kappa\Big(G_1, \boldsymbol{k}^{1ij}, \ell_1^{ij}\Big), i\right\} + \max\left\{\kappa\Big(G_2, \boldsymbol{k}^{2ij}, \ell_2^{ij}\Big), j\right\},$$

where

- $\boldsymbol{k}^{1ij} = \boldsymbol{k}|_{V_1} - \min\{i + j, n_2\} \cdot \mathbf{1}_{V_1}$,
- $\boldsymbol{k}^{2ij} = \boldsymbol{k}|_{V_2} - \min\{i + j, n_1\} \cdot \mathbf{1}_{V_2}$,
- $\ell_1^{ij} = \min\{\ell, i + j + 1\} - \min\{i + j, n_2\}$,
- $\ell_2^{ij} = \min\{\ell, i + j + 1\} - \min\{i + j, n_1\}$.

Proof. First, we show that $\min_{(i,j)\in\mathcal{F}} f(i, j) \leq \kappa(G, \boldsymbol{k}, \ell)$. Let $S \subseteq V$ be a minimum vector connectivity set for $(G, \boldsymbol{k}, \ell)$, that is, $|S| = \kappa(G, \boldsymbol{k}, \ell)$, S is a vector connectivity set for $(G, \boldsymbol{k})$, and $v \in S$ for all $v \in V$ such that $k_v \geq \ell$. Let $S_i = S \cap V_i$, for $i = 1, 2$, and let $I = |S_1|$ and $J = |S_2|$.

Claim 1. $\kappa\Big(G_1, \boldsymbol{k}^{1IJ}, \ell_1^{IJ}\Big) \leq I$

In order to prove Claim 1, it is enough to argue that S_1 is a vector connectivity set for $(G_1, \boldsymbol{k}^{1IJ}, \ell_1^{IJ})$, Equivalently, it suffices to prove that

(i) for every $v \in V_1 \setminus S_1$, there exists a v–S_1 fan in G_1 of order $k_v^{1IJ} = k_v - \min\{I + J, n_2\}$, and

(ii) for every $v \in V_1$ such that

$$k_v^{1IJ} \geq \ell_1^{IJ} = \min\{\ell, I + J + 1\} - \min\{I + J, n_2\},$$

we have $v \in S_1$.

Let us first show that condition (i) follows from the fact that S is a vector connectivity set for $(G, \boldsymbol{k})$. Indeed, for every $v \in V_1 \setminus S_1 = V_1 \setminus S$, there exists a v–S fan of order k_v in G. Let $\mathcal{P}$ be a v–S fan of maximum order in G that

minimizes the number of paths entirely contained in G_1, and, subject to this condition, is of smallest total path length. Then $\mathcal{P}$ contains all J one-edge paths of the form (v, x) where $x \in S_2$. Moreover, $\mathcal{P}$ contains as many two-edge paths of the form (v, x, y) as possible, where $x \in V_2 \setminus S_2$ and $y \in S_1$; this number of paths is equal to $\min\{I, n_2 - J\}$. Every other path of $\mathcal{P}$ is entirely contained in G_1. Hence, the number of paths in $\mathcal{P}$ entirely contained in G_1 is equal to $|\mathcal{P}| - J - \min\{I, n_2 - J\}$, which is at least $k_v - \min\{I + J, n_2\} = k_v^{1IJ}$. This shows that condition (i) holds.

To show (ii), suppose that $v \in V_1$ is a vertex with $k_v^{1IJ} \geq \ell_1^{IJ}$, that is,

$$k_v - \min\{I + J, n_2\} \geq \min\{\ell, I + J + 1\} - \min\{I + J, n_2\},$$

or, equivalently,

$$k_v \geq \min\{\ell, I + J + 1\}. \tag{4}$$

We want to show that $v \in S_1$. Suppose, for contradiction, that $v \in V_1 \setminus S_1$. Since S is a vector connectivity set for $(G, \boldsymbol{k}, \ell)$ and $v \notin S$, we have $k_v \leq \ell - 1$ and $k_v \leq |S| = I + J$. Therefore, $k_v \leq \min\{\ell - 1, I + J\} = \min\{\ell, I + J + 1\} - 1$, contradicting (4). This shows that $v \in S_1$.

With an analogous argument, one can prove the following claim.

Claim 2. $\kappa\Big(G_2, \boldsymbol{k}^{2IJ}, \ell_2^{IJ}\Big) \leq J$

Claims 1 and 2 together imply that $f(I, J) = I + J$, and consequently $\min_{(i,j)\in\mathcal{F}} f(i, j) \leq f(I, J) = I + J = |S| = \kappa(G, \boldsymbol{k}, \ell)$.

Now we show that $\kappa(G, \boldsymbol{k}, \ell) \leq \min_{(i,j)\in\mathcal{F}} f(i, j)$. Let (I, J) be a pair from $\mathcal{F}$ such that $f(I, J) = \min_{(i,j)\in\mathcal{F}} f(i, j)$. Write $\min_{(i,j)\in\mathcal{F}} f(i, j) = m_1 + m_2$, where

$$m_1 = \max\left\{\kappa\Big(G_1, \boldsymbol{k}^{1IJ}, \ell_1^{IJ}\Big), I\right\}$$

and

$$m_2 = \max\left\{\kappa\Big(G_2, \boldsymbol{k}^{2IJ}, \ell_2^{IJ}\Big), J\right\}.$$

Let S_1' and S_2' be minimum vector connectivity sets for $(G_1, \boldsymbol{k}^{1IJ}, \ell_1^{IJ})$ and $(G_2, \boldsymbol{k}^{2IJ}, \ell_2^{IJ})$, respectively. For $i = 1, 2$, let $S_i \subseteq V_i$ be a set of size m_i

containing S'_i. Set $S = S_1 \cup S_2$. To show that $\kappa(G, \boldsymbol{k}, \ell) \leq \min_{(i,j)\in\mathcal{F}} f(i,j)$, it suffices to show that S is a vector connectivity set for $(G, \boldsymbol{k}, \ell)$. Equivalently, it suffices to prove that

(I) for every $v \in V \setminus S$, there exists a v–S fan in G of order k_v, and

(II) for every $v \in V$ such that $k_v \geq \ell$, we have $v \in S$.

Let $v \in V \setminus S$. Suppose that $v \in V_1 \setminus S_1$. Let $\mathcal{P}$ be a v–S_1 fan in G_1 of order $k_v^{1IJ} = k_v - \min\{I + J, n_2\}$. Then, $\mathcal{P}$ is also a v–S fan in G. Extend $\mathcal{P}$ to a larger v–S fan in G by adding to it all m_2 one-edge paths of the form (v, x) where $x \in S_2$, and as many two-edge paths of the form (v, x, y) as possible, where $x \in V_2 \setminus S_2$ and $y \in S_1$; the number of these two-edge paths is equal to $\min\{n_2 - m_2, m_1 - k_v^{1IJ}\}$. The total number of paths in the so constructed v–S fan is equal to

$$\begin{aligned} |\mathcal{P}| + m_2 + \min\{n_2 - m_2, m_1 - k_v^{1IJ}\} &= \min\{k_v^{1IJ} + n_2, m_1 + m_2\} \\ &\geq \min\{k_v^{1IJ} + n_2, I + J\}\,. \end{aligned}$$

We would like to show that $\min\{k_v^{1IJ} + n_2, I + J\} \geq k_v$, or, equivalently, that

$$k_v^{1IJ} + n_2 \geq k_v \tag{5}$$

and

$$I + J \geq k_v\,. \tag{6}$$

Inequality (5) follows directly from the definition of k_v^{1IJ}. Since S_1 is a vector connectivity set for $(G_1, \boldsymbol{k}^{1IJ}, \ell_1^{IJ})$ and $v \in V_1 \setminus S_1$, it follows that

$$k_v^{1IJ} \leq \ell_1^{IJ} - 1\,,$$

that is,

$$k_v - \min\{I + J, n_2\} \leq \min\{\ell - 1, I + J\} - \min\{I + J, n_2\}\,,$$

or, equivalently,

$$k_v \leq \min\{\ell - 1, I + J\}\,.$$

In particular, comparing k_v to the second term in the above minimum yields the desired inequality (6). This shows that for every $v \in V_1 \setminus S_1$, there exists a v–S fan in G of order k_v. We can show similarly that for every $v \in V_2 \setminus S_2$, there exists a v–S fan in G of order k_v. This shows (I).

To show (II), let $v \in V$ be such that $k_v \geq \ell$. We need to show that $v \in S$. Without loss of generality, assume that $v \in V_1$. The condition $k_v \geq \ell$ implies that

$$k_v^{1IJ} = k_v - \min\{I + J, n_2\} \geq \ell - \min\{I + J, n_2\} \geq \ell_1^{IJ}\,,$$

which implies that $v \in S_1$ due to the fact that S_1 is a vector connectivity set for $(G_1, \boldsymbol{k}^{1IJ}, \ell_1^{IJ})$. Hence, $v \in S$. This shows that condition (II) holds, and hence that $\kappa(G, \boldsymbol{k}, \ell) \leq \min_{(i,j)\in\mathcal{F}} f(i,j)$. We conclude that $\kappa(G, \boldsymbol{k}, \ell) = \min_{(i,j)\in\mathcal{F}} f(i,j)$.

□

Theorem 3. Vector Connectivity *can be solved in polynomial time on cographs.*

Proof. Consider the input $(G = (V, E), \boldsymbol{k})$ to the Vector Connectivity problem, where G is a cograph and $n = |V|$. By Lemma 5, computing the value of $\kappa(G, \boldsymbol{k})$ is equivalent to computing the value of $\kappa(G, \boldsymbol{k}, K)$ with $K = \max_{v\in V} k_v + 1$. We compute this value as follows. First, we compute a nice cotree T of G. We traverse T bottom up, processing a node only after all its children have been processed. When processing a node t of T, we compute all $O(n^2)$ values of

$$\kappa(H, \boldsymbol{k}|_{V(H)} - i \cdot \mathbf{1}_{V(H)}, \ell)\,, \quad i \in \{0, 1, \ldots, n\}, \quad \ell \in \{0, 1, \ldots, K\}\,,$$

where H is the induced subgraph of G corresponding to the subtree T_t. For every leaf of the cotree, corresponding to a single vertex v of G, each of the $O(n^2)$ values can be computed in $O(1)$ time as follows:

$$\kappa((\{v\}, \emptyset), k_v - i, \ell) = \begin{cases} 0 & \text{if } k_v - i \leq \min\{\ell - 1, 0\}\,, \\ 1 & \text{otherwise.} \end{cases}$$

Depending on whether an internal node t is a $\oplus$-node or a $\otimes$-node, we can use Lemma 6 or Lemma 7 to compute each of the $O(n^2)$ values of $\kappa(H, \boldsymbol{k}|_{V(H)} - i \cdot \mathbf{1}_{V(H)}, \ell)$ in time $O(n^2)$. Hence, each internal node of the modified cotree can be processed in time $O(n^4)$, yielding an overall time complexity of $O(n^5)$, since a cotree has $O(n)$ nodes.

A minimum vector connectivity set can also be computed in the stated time. In addition to the values of $\kappa(H, \boldsymbol{k}|_{V(H)} - i \cdot \mathbf{1}_{V(H)}, \ell)$ at each node of the cotree, we need to store also a minimum vector connectivity set achieving each of these values. These sets can be computed recursively as follows. For an internal node t with corresponding subgraph H, let H_1 and H_2 denote the subgraphs of G corresponding to the two children of t in T.

- If H corresponds to a leaf of T, then $V(H) = \{v\}$ for some $v \in V$, and a minimum vector connectivity set for $(H, k_v - i, \ell)$ is either empty or $\{v\}$, depending on whether $k_v - i \leq \min\{\ell - 1, 0\}$ or not.

- If $H = H_1 \oplus H_2$, then a minimum vector connectivity set for $(H, \boldsymbol{k}|_{V(H)} - i \cdot \mathbf{1}_{V(H)}, \ell)$ is given by the union of minimum vector connectivity sets for $(H_1, \boldsymbol{k}|_{V(H_1)} - i \cdot \mathbf{1}_{V(H)}, \ell)$ and $(H_2, \boldsymbol{k}|_{V(H_2)} - i \cdot \mathbf{1}_{V(H)}, \ell)$.

- If $H = H_1 \otimes H_2$, then a minimum vector connectivity set S for $(H, \boldsymbol{k}|_{V(H)} - i \cdot \mathbf{1}_{V(H)}, \ell)$ can be computed in $O(n^2)$ time: first compute a pair (I, J) minimizing the function f defined in Lemma 7 (with H, H_1, H_2 in place of G, G_1, G_2, respectively), and then take the union of minimum vector connectivity sets S_1' and S_2' for $(H_1, \boldsymbol{k}^{1IJ}, \ell_1^{IJ})$ and $(H_2, \boldsymbol{k}^{2IJ}, \ell_2^{IJ})$, together with some extra vertices if necessary so that $|S \cap V(H_1)| \geq I$ and $|S \cap V(H_2)| \geq J$.

Finally, let us remark that all the instances $(H, \boldsymbol{k}', \ell)$ for which $\kappa(H, \boldsymbol{k}', \ell)$ must be evaluated in order to compute the value of $\kappa(G, \boldsymbol{k}) = \kappa(G, \boldsymbol{k}, \max_{v \in V} k_v + 1)$ satisfy the property that for all $v \in V(H)$, either $k_v' \leq d_H(v)$ or $k_v' \geq \ell$. This can be proved by induction on the distance of a node t representing H from the root of the cotree T, and assures that the val-

ues of $\kappa(H, \boldsymbol{k}', \ell)$ are well defined for such instances. This completes the proof of Theorem 3. □

6 A Polynomial-Time Algorithm for Trees

We have seen that Vector Connectivity is solvable in polynomial time on cographs and split graphs. These two graph classes do not contain graphs with long induced paths. In particular, cographs are equivalent to graphs that do not have induced paths of length 3 or more [6], and it is easy to observe that split graphs do not contain induced paths of length 4 or more. In this section, we give a polynomial-time algorithm to solve the Vector Connectivity problem on trees, a graph class that allows the existence of arbitrarily long induced paths.

Theorem 4. Vector Connectivity *can be solved in polynomial time on trees.*

Proof. Let $(T, \boldsymbol{k})$ be an instance of Vector Connectivity, where $T = (V, E)$ is a tree. We assume that T has at least two vertices and is rooted at an arbitrary vertex r. Since the requirements of the vertices do not change during the execution of the algorithm, we will simply speak of a vector connectivity set for T_v instead of a vector connectivity set for $(T_v, \boldsymbol{k}|_{V(T_v)})$, for every $v \in V$. Recall that for a given rooted tree T, we write T_v to denote the subtree rooted at vertex v, including vertex v.

The idea of the algorithm is to construct a vector connectivity set for T of minimum size, starting from the leaves of T and processing a vertex only after all its children have been processed. At any step of the algorithm, let $S \subseteq V$ be the set of vertices that have thus far been chosen to belong to the solution. For any vertex v of T, we define $S_v = S \cap V(T_v)$. When processing a vertex v, the algorithm computes the values $b(v)$, $c(v)$, and $n(v)$, where the letters b, c, and n stand for "below", "children", and "needs", respectively, as we will explain below. The functions b, c, and n are defined as follows. For every vertex $v \in V$, the value of $b(v)$ indicates whether or not there is a vertex of S in the

subtree "below" v. More precisely, $b(v) = 1$ if the subtree T_v contains at least one vertex of S, and $b(v) = 0$ otherwise. The value $c(v)$ denotes the number of children w of v for which $b(w) = 1$. Note that if $b(w) = 1$ for a child w of vertex v, then v is 1-connected but not 2-connected to S_w, regardless of how many vertices S_w contains. Furthermore, v is 1-connected to $S \setminus V(T_v)$ if S contains a vertex outside T_v. We let $n(v)$ denote whether or not a vertex in T_v "needs" an additional path to a vertex outside of T_v, indicated by 1 or 0, for every $v \in V$. More precisely, $n(v) = 0$ if for every vertex $w \in V(T_v)$, there is a w–S_v fan of order k_w in T_v, i.e., every vertex w of T_v, including v itself, is k_w-connected to S_v and hence also to S. On the other hand, $n(v) = 1$ if there is a vertex $w \in V(T_v)$ such that there is a w–S_v fan of order $k_w - 1$ but no w–S_v fan of order k_w in T_v.

We now describe the algorithm in detail. Initially, we set $S = \emptyset$. Let $v \in V$ be a leaf of T. We set $c(v) = 0$. If $k_v = 0$, then we set $b(v) = 0$ and $n(v) = 0$. If $k_v = 1$, then we set $b(v) = 0$ and $n(v) = 1$.

Next, let v be a vertex that is not a leaf and not the root, and assume that the children of v have all been processed. For every child w of v, if $n(w) = 1$ and v has a child $w' \neq w$ such that $b(w') = 1$, then we set $n(w) = 0$. We then compute $c(v)$ by adding up the b-values of all the children of v. If $k_v \leq c(v)$, then we set $n(v) = 1$ if v has a child w with $n(w) = 1$, and we set $n(v) = 0$ otherwise. If $k_v = c(v) + 1$, then we set $n(v) = 1$. If $k_v \geq c(v) + 2$, then we add v to S and set $n(v) = 0$. In each of the above cases, we set $b(v) = 1$ if $c(v) \geq 1$ or if v is added to S, and we set $b(v) = 0$ otherwise.

Finally, let v be the root of T. We set $n(v) = 0$. If $k_v \leq c(v)$, then we perform the following check: if v has a child w such that $n(w) = 1$ and $b(w') = 0$ for every other child $w' \neq w$ of v, then we add v to S. If $k_v \geq c(v)+1$, then we add v to S. The algorithm outputs the set S and terminates as soon as the root has been processed.

In order to prove the correctness of the algorithm, we prove that, for every $v \in V$, the following three statements are true immediately after v is processed, where p denotes the parent of v in T.

(i) If $n(v) = 0$, then S_v is a vector connectivity set for T_v.

(ii) If $n(v) = 1$, then $S_v \cup \{p\}$ is a vector connectivity set for the subtree of T induced by $V(T_v) \cup \{p\}$;

(iii) There is no vector connectivity set S' for T such that $|S' \cap V(T_v)| < |S_v|$.

Assume first that v is a leaf. Then $d_T(v) = 1$, so $k_v \in \{0, 1\}$. Note that the algorithm does not add v to S, regardless of whether $k_v = 0$ or $k_v = 1$. Hence $S_v = \emptyset$, and statement (iii) trivially holds in both cases. If $k_v = 0$, then we set $n(v) = 0$, and it is clear that all statements are satisfied in this case. If $k_v = 1$, then we set $n(v) = 1$. Since the set $\{p\}$ consisting of the parent of v is a vector connectivity set for the subtree of T induced by $V(T_v) \cup \{p\}$, statement (ii) holds.

Assume now that v is not a leaf and not the root. We distinguish two cases, depending on whether or not v is added to S by our algorithm. Suppose v is added to S. Then $k_v \geq c(v)+2$, and we set $n(v) = 0$ in this case. Since v is added to S, we have that every descendant w of v with $n(w) = 1$ becomes k_w-connected to S_v, so statement (i) holds. To see why statement (iii) holds in this case, let S' be any vector connectivity set for T. We know that $|S' \cap V(T_w)| \geq |S_w|$ for every child w of v, since statement (iii) holds for every child w. Hence, if $v \in S'$, then $|S' \cap V(T_v)| \geq |S_v|$ and statement (iii) holds. If $v \notin S'$, then the fact that $k_v \geq c(v)+2$ implies that S' must contain at least one vertex of a subtree T_w for a child w of v with $b(w) = 0$. Hence we have $|S' \cap V(T_v)| \geq |S_v|$ and statement (iii) is true also in this case.

Now suppose that v is not added to S. Since statement (iii) holds for all children of v and $v \notin S$, statement (iii) is trivially true for v. Let us argue why statements (i) and (ii) hold. If $n(v) = 0$, then for every child w with $n(w) = 1$ just before v is processed there is another child $w' \neq w$ with $b(w') = 1$. Hence every child w with $n(w) = 1$ just before v is processed is k_w-connected to S, since it has a w–S_v fan of order $k_w - 1$ and one additional path from w via v to a vertex of $S \cap V(T_{w'})$ for some child $w' \neq w$ with $b(w') = 1$; the same holds for each descendant w'' of w with $n(w'') = 1$. Hence S_v is a vector connectivity

set for T_v, and statement (i) holds in this case. If $n(v) = 1$ and $k_v \leq c(v)$, then at least one vertex $w \in V(T_v)$ is $(k_w - 1)$-connected to S_v (in fact, to S_w), but needs a path from w to a vertex of S outside T_v to become k_w-connected to S. Clearly, adding the parent p of v to S_v ensures that all such vertices w become k_w-connected to $S_v \cup \{p\}$. If $n(v) = 1$ and $k_v = c(v) + 1$, then every vertex w in T_v with $n(w) = 1$ that is $(k_w - 1)$-connected to S_v becomes k_w-connected to $S_v \cup \{p\}$. In either case, statement (ii) holds.

Finally, assume that v is the root of T. Then we set $n(v) = 0$, so we need to show that statement (i) holds. This is clearly the case if v is added to S, since then every descendant w of v with $n(w) = 1$ becomes k_w-connected to S_v. If v is not added to S, then $k_v \leq c(v)$ and for every child w of v with $n(w) = 1$, there is another child $w' \neq w$ such that $b(w') = 1$ and consequently there is a path from w via v to a vertex of S outside T_w. Hence $S_v = S$ is a vector connectivity set for $T_v = T$, and statement (i) also holds in this case. If v was not added to S, then the validity of statement (iii) immediately follows from the fact that statement (iii) holds for every child of v. Suppose v was added to S, and let S' be any vector connectivity set for T. Since statement (iii) holds for any child w of v, we have $|S' \cap V(T_w)| \geq |S_w|$. Just as in the case where v is not a leaf and not the root, we have $|S' \cap V(T_v)| \geq |S_v|$ regardless of whether or not $v \in S'$. Hence statement (iii) is true also in this case.

Since these statements hold for the root of T, the set S constructed by the algorithm is a vector connectivity set for T of minimum size. The observation that all steps of the algorithm can be performed in polynomial time completes the proof of Theorem 4. □

7 Concluding Remarks

In this paper, we initiated the study of the Vector Connectivity problem, which opens a research path with many interesting questions. The most prominent of these questions is of course the computational complexity of Vector Connectivity on general input graphs. Could it be that the problem is

polynomial-time solvable on all graphs, or is its tractability heavily dependent on either the absence of long induced paths or on a tree-like structure of the input graph? On which other graph classes is Vector Connectivity solvable in polynomial time? Is the problem polynomial-time solvable for all graphs with a constant bound on the length of induced paths? Does Vector Connectivity admit a polynomial-time constant-factor approximation algorithm on general graphs?

Another interesting variant of the problem can be obtained by allowing the requirement k_v of each vertex v to be arbitrarily large, in which case a vertex v with $k_v > d_G(v)$ is forced to be in every vector connectivity set. Is it perhaps easier to prove this variant to be NP-hard in general? Note that the algorithms given in this paper, except the algorithm for split graphs, work in polynomial time also for this variant. Is this variant polynomial-time solvable on split graphs?

Note added in proof: Since the submission of this paper, Romeo Rizzi proved that the Vector Connectivity problem is NP-hard on general graphs.

Acknowledgements

Part of this work was carried out while the fourth author was visiting the second and third authors at the Department of Informatics of University of Bergen. Their hospitality is gratefully acknowledged. We are indebted to Ferdinando Cicalese and Ugo Vaccaro for initial discussions during which the Vector Connectivity problem was discovered and research on it started. We are also grateful to Nicola Apollonio for helpful discussions, and to the two anonymous referees for their comments. In particular, one of the referees suggested a simplification in the proof of Lemma 3, and the other one the open problem of whether the vector connectivity problem is polynomially solvable on all graphs without long induced paths.

This work is supported by the Research Council of Norway (197548/F20) and by the Slovenian Research Agency (research program P1–0285 and re-

search projects J1–4010, J1–4021, BI-US/12–13–029 and N1–0011: GReGAS, supported in part by the European Science Foundation).

References

[1] E. Boros, P. Heggernes, P. van 't Hof, and M. Milanič, Vector connectivity in graphs, Proc 10th Annual Conf on Theory and Applications of Models of Computation, Lecture Notes in Computer Science, Vol. 7876, Springer, 2013, pp. 331–342.

[2] N. Chen, On the approximability of influence in social networks. SIAM J Discrete Math 23 (2009), 1400–1415.

[3] M. Chlebík and J. Chlebíkova, Approximation hardness of dominating set problems in bounded degree graphs, Inform and Comput 206 (2008), 1264–1275.

[4] F. Cicalese, M. Milanič, and U. Vaccaro, On the approximability and exact algorithms for vector domination and related problems in graphs, Discr Appl Math 161 (2013), 750–767.

[5] D.G. Corneil, H. Lerchs, and L. Stewart Burlingham, Complement reducible graphs, Discr Appl Math 3 (1981), 163–174.

[6] D.G. Corneil, Y. Perl, and L.K. Stewart, A linear recognition algorithm for cographs, SIAM J Comput 14 (1985), 926–934.

[7] M.R. Garey and D.S. Johnson, Computers and intractability, W.H. Freeman and Co., New York, 1979.

[8] M.C. Golumbic, Algorithmic graph theory and perfect graphs, Second edition, Ann Discrete Math 57, Elsevier, 2004.

[9] M. Habib and C. Paul, A simple linear time algorithm for cograph recognition, Discr Appl Math 145 (2005), 183–197.

[10] P.L. Hammer and B. Simeone, The splittance of a graph, Combinatorica 1 (1981), 275–284.

[11] J. Harant, A. Prochnewski, and M. Voigt, On dominating sets and independent sets of graphs, Combin Probab Comput 8 (1999), 547–553.

[12] T.W. Haynes, S.T. Hedetniemi, and P.J. Slater, Fundamentals of domination in graphs, Dekker, New York, 1998.

[13] T.W. Haynes, S.T. Hedetniemi, and P.J. Slater, Domination in graphs: Advanced topics, Marcel Dekker, New York, 1998.

[14] D. Kempe, J. Kleinberg, and E. Tardos, Maximizing the spread of influence through a social network, Proc 9th ACM SIGKDD Intl Conf on Knowledge Discovery and Data Mining, ACM Press, 2003, pp. 137–146.

[15] N. Mahadev and U. Peled, Threshold graphs and related topics, Ann Discrete Math 56, North Holland, 1995.

[16] A. Nichterlein, R. Niedermeier, J. Uhlmann, and M. Weller, On tractable cases of Target Set Selection, Soc Netw Anal Min 3 (2013), 233–256.

[17] J. Oxley, Matroid theory, Oxford University Press, Oxford, 1992.

[18] H. Perfect, Applications of Menger's graph theorem, J Math Anal Appl 22 (1968), 96–111.

[19] J.S. Pym, A proof of the linkage theorem, J Math Anal Appl 27 (1969), 636–638.

[20] A. Schrijver, Combinatorial optimization. Polyhedra and efficiency. Vol. A–C. Algorithms and Combinatorics, 24. Springer-Verlag, Berlin, 2003.

[21] D.J.A. Welsh, Matroid theory, L.M.S. Monographs, 8. Academic Press, London-New York, 1976.

[22] L.A. Wolsey, An analysis of the greedy algorithm for the submodular set covering problem, Combinatorica 2 (1982), 385–393.