On Kernelization and Approximation for the Vector Connectivity Problem*

Stefan Kratsch[1] and Manuel Sorge[2]

1 **University of Bonn, Germany**
`kratsch@cs.uni-bonn.de`

2 **Technical University Berlin, Germany**
`manuel.sorge@tu-berlin.de`

Abstract

In the VECTOR CONNECTIVITY problem we are given an undirected graph $G = (V, E)$, a demand function $\phi\colon V \to \{0, \ldots, d\}$, and an integer k. The question is whether there exists a set S of at most k vertices such that every vertex $v \in V \setminus S$ has at least $\phi(v)$ vertex-disjoint paths to S; this abstractly captures questions about placing servers in a network, or warehouses on a map, relative to demands. The problem is NP-hard already for instances with $d = 4$ (Cicalese et al., Theor. Comput. Sci. '15), admits a log-factor approximation (Boros et al., Networks '14), and is fixed-parameter tractable in terms of k (Lokshtanov, unpublished '14).

We prove several results regarding kernelization and approximation for VECTOR CONNECTIVITY and the variant VECTOR d-CONNECTIVITY where the upper bound d on demands is a constant. For VECTOR d-CONNECTIVITY we give a factor d-approximation algorithm and construct a vertex-linear kernelization, i.e., an efficient reduction to an equivalent instance with $f(d)k = \mathcal{O}(k)$ vertices. For VECTOR CONNECTIVITY we get a factor opt-approximation and we show that it has no kernelization to size polynomial in $k + d$ unless $\mathsf{NP} \subseteq \mathsf{coNP/poly}$, making $f(d)\,\mathrm{poly}(k)$ optimal for VECTOR d-CONNECTIVITY. Finally, we provide a write-up for fixed-parameter tractability of VECTOR CONNECTIVITY(k) by giving a different algorithm based on matroid intersection.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases parameterized complexity, kernelization, approximation

Digital Object Identifier 10.4230/LIPIcs.IPEC.2015.377

1 Introduction

In the VECTOR CONNECTIVITY problem we are given an undirected graph $G = (V, E)$, a demand function $\phi\colon V \to \{0, \ldots, d\}$, and an integer $k \in \mathbb{N}$. The question is whether there exists a set S of at most k vertices of G such that every vertex $v \in V$ is either in S or has at least $\phi(v)$ vertex-disjoint paths to vertices in S; the paths may share the vertex v itself.

> VECTOR CONNECTIVITY
> **Input:** A graph $G = (V, E)$, a function $\phi\colon V \to \{0, \ldots, d\}$, $k \in \mathbb{N}$.
> **Question:** Is there a set S of at most k vertices such that each vertex $v \in V \setminus S$ has $\phi(v)$ vertex-disjoint paths with endpoints in S?

The value $\phi(v)$ is also called the *demand* of vertex v. We call $S \subseteq V$ a *vector connectivity set* for (G, ϕ), if it fulfills the requirements above. We do not formally distinguish decision and optimization version; k is not needed for the latter.

* Supported by the German Research Foundation (DFG) under grants KR 4286/1-1 and NI 369/12-2.

For intuition about the problem formulation and applications one may imagine a logistical problem about placing warehouses to service locations on a map (or servers in a network): Each location has a particular demand, which could capture volume and redundancy requirements or level of importance. Demand can be satisfied by placing a warehouse at the location or by ensuring that there are enough disjoint paths from the location to warehouses; both can be seen as guaranteeing sufficient throughput or ensuring access that is failsafe up to demand minus one interruptions in connections. In this way, VECTOR CONNECTIVITY can also be seen as a variant of the FACILITY LOCATION problem in which the costs of serving demand are negligible, but instead redundancy is required.

Related work. The study of the VECTOR CONNECTIVITY problem was initiated recently by Boros et al. [2] who gave polynomial-time algorithms for trees, cographs, and split graphs. Moreover, they obtained an $\ln n + 2$-factor approximation algorithm for the general case. More recently, Cicalese et al. [3] continued the study of VECTOR CONNECTIVITY and amongst other results proved that it is APX-hard (and NP-hard) on general graphs, even when all demands are upper bounded by four. In a related talk during a recent Dagstuhl seminar (Dagstuhl Seminar 14071 on "Graph Modification Problems.") Milanič asked whether VECTOR CONNECTIVITY is fixed-parameter tractable with respect to the maximum solution size k. This was answered affirmatively by Lokshtanov (unpublished).

Our results. We obtain results regarding kernelization and approximation for VECTOR CONNECTIVITY and VECTOR d-CONNECTIVITY where the maximum demand d is a fixed constant. We also provide a self-contained write-up for fixed-parameter tractability of VECTOR CONNECTIVITY with parameter k (Sec. 4); it is different from Lokshtanov's approach and instead relies on a matroid intersection algorithm of Marx [8].

Our analysis of the problem starts with a new data reduction rule stating that we can safely "forget" the demand of $r := \phi(v)$ at a vertex v if v has vertex-disjoint paths to r vertices each of demand at least r (Sec. 2). After exhaustive application of the rule, all remaining vertices of demand r must have cuts of size at most $r - 1$ separating them from other vertices of demand at least r. By analyzing these cuts we then show that any yes-instance of VECTOR d-CONNECTIVITY can have at most d^2k vertices with nonzero demand; the corresponding bound for VECTOR CONNECTIVITY is $k^3 + k$. (Both bounds also hold when replacing k by the optimum cost opt). This directly would yield factor d^2 and factor $\mathsf{opt}^2 + 1$ approximation algorithms. We improve upon this in Sec. 3 by giving a variant of the reduction rule that works correctly relative to any partial solution S_0, which can then be applied in each round of our approximation algorithm. The algorithm follows the local-ratio paradigm and, surprisingly perhaps, proceeds by always selecting a vertex of *lowest* demand. We thus obtain ratios of d and opt respectively, i.e., the returned solution is of size at most $d \cdot \mathsf{opt}$ for VECTOR d-CONNECTIVITY and at most opt^2 for VECTOR CONNECTIVITY.

Regarding kernelization we show in Sec. 6 that there is no kernel with size polynomial in k or even $k + d$ for VECTOR CONNECTIVITY unless $\mathsf{NP} \subseteq \mathsf{coNP/poly}$ (and the polynomial hierarchy collapses); our proof also implies that the problem is WK[1]-hard (cf. [6]). Nevertheless, when d is a fixed constant, we prove that a vertex-linear kernelization is possible. A non-constructive proof of this can be pieced together from recent work on meta kernelization on sparse graph classes (see below). Instead we give a constructive algorithm building on an explicit (though technical) description of what constitutes equivalent subproblems. We also have a direct proof for the number of such subproblems in an instance with parameter k rather than relying on a known argument for bounding the number of connected subgraphs of bounded size and bounded neighborhood size (via the two-families theorem of Bollobas, cf. [7]). A brief overview of our kernelization is given in Sec. 5. The bound of $f(d)k = \mathcal{O}(k)$ vertices is optimal in the sense that the lower bound for parameter $d + k$ rules out total size poly$(k + d)$, i.e. we need to allow superpolynomial dependence on d.

A non-constructive kernelization argument. The mentioned results on meta kernelization for problems on sparse graph classes mostly rely on the notion of a protrusion, i.e., an induced subgraph of bounded treewidth such that only a bounded number of its vertices have neighbors in the rest of the graph (the *boundary*). Under appropriate assumptions there are general results that allow the replacements of protrusions by equivalent ones of bounded size, which yields kernelization results assuming that the graph can be decomposed into a bounded number of protrusions. Intuitively, having small boundary size limits the interaction of a protrusion with the remaining graph. The assumption of bounded treewidth ensures fast algorithms for solving subproblems on the protrusion and also leads to fairly general arguments for obtaining small equivalent replacements. Note that for VECTOR CONNECTIVITY there is no reason to assume small treewidth and we also do not restrict the input graphs.

Arguably, the most crucial properties of a protrusion are the small boundary and the fact that we can efficiently compute subproblems on the protrusion; in principle, we do not need bounded treewidth. (Intuitively, we want to know the best solution value for each choice of interaction of a global solution with the boundary of the protrusion.) Thus, it seems natural to define a relaxed variant of protrusions by insisting on a small boundary and efficient algorithms for solving subproblems rather than demanding bounded treewidth. Fomin et al. [5] follow this approach for problems related to picking a certain subset of vertices like DOMINATING SET: Their relaxed definition of a r-DS-protrusion requires a boundary of size at most r and the existence of a solution of size at most r for the protrusion itself; the latter part implies efficient algorithms since we can afford to simply try all $r = \mathcal{O}(1)$ sized vertex subsets. Fomin et al. also derive a general protrusion replacement routine for problems that have finite integer index (a common assumption for meta kernelization, see, e.g., Fomin et al. [5]) and are monotone when provided also with a sufficiently fast algorithm, like the one implied by having a solution of size at most r for the protrusion. Fomin et al. [5] remark that the procedure is not constructive since it assumes hard-wiring an appropriate set of representative graphs, whose existence is implied by being finite integer index.

From previous work of Cicalese [3] it is known that VECTOR CONNECTIVITY is an implicit hitting set problem where the set family consists of all connected subgraphs with neighborhood size smaller than the largest demand in the set; the family is exponential size, but we get size roughly $\mathcal{O}(n^d)$ when demands are at most d. The procedure of Fomin et al. [5] can be applied to minimal sets in this family and will shrink them to some size bounded by an unknown function in d, say $h(d)$. Then one can apply the two-families theorem of Bollobas (cf. [7]) to prove that each vertex is contained in at most $\binom{h(d)+d}{d}$ such sets. Because the solution must hit all sets with k vertices, a yes-instance can have at most $k \cdot \binom{h(d)+d}{d}$ sets. This argument can be completed to a vertex-linear kernelization.

In comparison, we obtain an explicit upper bound of $d^2k \cdot 2^{d^3+d}$ for the number of subproblems that need to be replaced, by considering a set family that is different from the implicit hitting set instance (but contains supersets of all those sets). We also have a constructive description of what constitutes equivalent subproblems. This enables us to give a single algorithm that works for all values of d based on maximum flow computations (rather than requiring for each value of d an algorithm with hard-wired representative subproblems).

Preliminaries. We use standard graph notation. Apart from this, due to the nature of the VECTOR CONNECTIVITY problem, we are frequently interested in disjoint paths from a vertex v to some vertex set S, where the paths are vertex-disjoint except for sharing v. The natural counterpart, in the spirit of Menger's theorem, are v, S*-separators* $C \subseteq V(G) \setminus \{v\}$ such that in $G - C$ no vertex of $S \setminus C$ is reachable from v; the maximum number of disjoint paths from v to S that may overlap in v equals the minimum size of a v, S-separator. Throughout, by disjoint paths from v to S, or v, S-separator (for any single vertex v and any vertex set S) we mean the mentioned path packings and separators with special role of v. Many proofs use the function $f\colon 2^V \to \mathbb{N}\colon U \mapsto |N(U)|$, which is well-known to be *submodular*, i.e., for all $X, Y \subseteq V$ we have $f(X) + f(Y) \geq f(X \cap Y) + f(X \cup Y)$.

So-called *closest sets* will be used frequently; these occur naturally in cut problems but appear to have no generalized name. We define a vertex set C to be *closest to* v if C is the unique v, C-separator of size at most $|C|$, where v, C-separator is in the above sense. As an example, if C is a minimum s, t-vertex cut that, amongst such cuts, has the smallest connected component for s in $G - C$ then C is also closest.

2 Reducing the number of demand vertices

In this section we introduce a reduction rule for VECTOR CONNECTIVITY that reduces the total demand. We prove that the reduction rule does not affect the solution space, which makes it applicable not only for kernelization but also for approximation and other techniques. In Section 5, we will use this rule in our kernelization for VECTOR d-CONNECTIVITY(k). In the following two sections, as applications of these reduction rules and the insights gained we get approximation algorithms for VECTOR d-CONNECTIVITY and VECTOR CONNECTIVITY, and an alternative FPT algorithm for VECTOR CONNECTIVITY(k).

▶ **Rule 1.** *Let (G, ϕ, k) be an instance of* VECTOR CONNECTIVITY. *If a vertex $v \in V$ has at least $\phi(v)$ vertex-disjoint paths to vertices different from v with demand at least $\phi(v)$ then set the demand of v to zero.*

We prove that the rule does not affect the space of feasible solutions.

▶ **Lemma 2.** *Let (G, ϕ, k) be an instance of* VECTOR CONNECTIVITY *and let (G, ϕ', k) be the instance obtained via a single application of Rule 1. Every $S \subseteq V(G)$ is a solution for (G, ϕ, k) if and only if it is a solution for (G, ϕ', k).*

Proof. Let v denote the vertex whose demand was set to zero by the reduction rule and define $r := \phi(v)$. Clearly, $\phi(u) = \phi'(u)$ for all vertices $u \in V(G) \setminus \{v\}$, and $\phi'(v) = 0$. It suffices to show that if S fulfills demands according to ϕ' then S fulfills also demands according to ϕ since $\phi(u) \geq \phi'(u)$ for all $u \in V(G)$. This in turn comes down to proving that S fulfills the demand of r at v assuming that it fulfills demands according to ϕ'. If $v \in S$ then the demand at v is trivially fulfilled so henceforth assume that $v \notin S$.

Let $w_1, \ldots, w_r$ denote vertices different from v with demand each at least r such that there exist r vertex-disjoint paths from v to $\{w_1, \ldots, w_r\}$, i.e., a single path to each w_i. Existence of such vertices is required for the application of the rule.

Assume for contradiction that S does not satisfy the demand of r at v (recall that $v \notin S$, by assumption), i.e., that there are no r vertex-disjoint paths from v to S that overlap only in v. It follows directly that there is a v, S-separator C of size at most $r - 1$. (Recall that C may contain vertices of S but not the vertex v.) Let R denote the connected component of v in $G - C$, then the following holds for each vertex $w_i \in \{w_1, \ldots, w_r\}$:

1. If $w_i \in S$ then $w_i \notin R$: Otherwise, we would have $S \cap R \supseteq \{w_i\} \neq \emptyset$ contradicting the fact that v can reach no vertex of S in $G - C$.
2. If $w_i \notin S$ then $w_i \notin R$: Since S fulfills demands according to ϕ' there must be at least r vertex-disjoint paths from w_i to S that overlap only in w_i. However, since $w_i \in R$ the set C is also a w_i, S-separator; a contradiction since C has size less than r.

Thus, no vertex from $w_1, \ldots, w_r$ is contained in R. This, however, implies that C separates v from $\{w_1, \ldots, w_r\}$, contradicting the fact that there are r vertex-disjoint paths from v to $\{w_1, \ldots, w_r\}$ that overlap only in v. It follows that no such v, S-separator C can exist, and, hence, that there are at least $r = \phi(v)$ vertex-disjoint paths from v to S, as claimed. Thus, S fulfills the demand of r at v and hence all demand according to ϕ. (Recall that the converse is trivial since $\phi(u) \geq \phi'(u)$ for all vertices $u \in V(G)$.) ◀

The idea for applying Rule 1 in polynomial time is to use a maximum-flow computation with v as source and all other vertices with nonzero demand as sinks.

To analyze the impact of Rule 1 we will now bound the number of nonzero demand vertices in an exhaustively reduced instance in terms of the optimum solution size opt and the maximum demand d. To this end, we require the technical lemma below about the structure of reduced instances as well as some notation. If (G, ϕ, k) is reduced according to Rule 1 then for each vertex v with demand $r = \phi(v) \geq 1$ there is a cut set C of size at most $r - 1$ that separates v from all other vertices with demand at least r. We fix for each vertex v with demand at least one a vertex set C, denoted $C(v)$, by picking the unique closest minimum v, D_v-separator where $D_v = \{u \in V \setminus \{v\} \mid \phi(u) \geq \phi(v)\}$. Furthermore, for such vertices v, let $R(v)$ denote the connected component of v in $G - C(v)$.

Clearly, a solution S must intersect each $R(v)$ since $|C(v)| < \phi(v)$. The following lemma shows implicitly that Rule 1 limits the amount of overlap of sets $R(v)$. Intuitively, sets $R(v)$ must overlap in order to "share" solution vertices.

▶ **Lemma 3.** *Let (G, ϕ, k) be reduced under Rule 1. Let $u, v \in V(G)$ be distinct vertices with $\phi(u) = \phi(v) \geq 1$. If $R(u) \cap R(v) \neq \emptyset$ then $u \in C(v)$ or $v \in C(u)$.*

Now, we give the promised bound on the number of nonzero demand vertices.

▶ **Lemma 4.** *Let (G, ϕ, k) be an instance of* VECTOR CONNECTIVITY *that is reduced under Rule 1 and let* opt *denote the minimum size of feasible solutions $S \subseteq V$. Then there are at most d^2*opt *nonzero demand vertices in G.*

Lemma 4 directly implies reduction rules for VECTOR d-CONNECTIVITY(k) and VECTOR CONNECTIVITY(k): For the former, if there are more than d^2k vertices then opt must exceed k and we can safely reject the instance. For the latter, there can be at most k vertices of demand greater than k since those must be in the solution. Additionally, if opt $\leq k$ then there are at most d^2opt $\leq k^3$ vertices of demand at most $d = k$; a total of $k^3 + k$.

We spell out the rule for VECTOR d-CONNECTIVITY(k) because it is used in our kernelization. The bound of $k^3 + k$ for VECTOR CONNECTIVITY(k) is crucial for our FPT-algorithm.

▶ **Rule 5.** *Let (G, ϕ, k) be reduced under Rule 1, with $\phi\colon V(G) \to \{0, \ldots, d\}$. If there are more than d^2k vertices of nonzero demand return* NO.

3 Approximation algorithm

In this section we discuss the approximability of VECTOR d-CONNECTIVITY. We know from Lemma 4 that the number of vertices with nonzero demand is at most d^2opt where opt denotes the minimum size solution for the instance in question. This directly implies a factor d^2-approximation because taking all nonzero demand vertices constitutes a feasible solution. We now show how to improve this to a factor d-approximation for VECTOR d-CONNECTIVITY.

The approximation algorithm will work as follows: We maintain an initially empty partial solution $S_0 \subseteq V$. In each round, we add at most d vertices to S_0 and show that this always brings us at least one step closer to a solution, i.e., the optimal number of required additional vertices shrinks by at least one. To achieve this, we update Rule 1 to take the partial solution S_0 into account.

▶ **Rule 6.** *Let (G, ϕ, k) be an instance of* VECTOR CONNECTIVITY *and let $S_0 \subseteq V(G)$. If there is a vertex v with non-zero demand and a vertex set W not containing v such that each vertex in W has demand at least $\phi(v)$ and v has at least $\phi(v)$ vertex-disjoint paths to $S_0 \cup W$, then set the demand of v to zero. Similarly, if $v \in S_0$ then also set its demand to zero.*

Intuitively, vertices in S_0 get the same status as vertices with demand at least $\phi(v)$ for applying the reduction argument. The proof of correctness now has to take into account that we seek a solution that includes S_0.

▶ **Lemma 7.** *Let (G, ϕ, k) be an instance of* VECTOR CONNECTIVITY, *let $S_0 \subseteq V(G)$, and let (G, ϕ', k) be the instance obtained via a single application of Rule 1. For every $S \subseteq V(G)$ it holds that $S \cup S_0$ is a solution for (G, ϕ, k) if and only if $S \cup S_0$ is a solution for (G, ϕ', k).*

It follows, that we can safely apply Rule 6, as a variant of Rule 1, in the presence of a partial solution S_0. It is easy to see that Rule 6 can be applied exhaustively in polynomial time because testing for any vertex v is a single two-way min-cut computation and each successful application lowers the number of nonzero demand vertices by one.

We now describe our approximation algorithm. The algorithm maintains an instance (G, ϕ), a set $S_0 \subseteq V(G)$, and an integer $\ell \in \mathbb{N}$. Given an instance (G, ϕ) the algorithm proceeds in rounds to build S_0, which will eventually be a complete (approximate) solution. We start with $S_0 = \emptyset$ and $\ell = 0$. In any single round, for given (G, ϕ), set S_0, and integer ℓ the algorithm proceeds as follows:

1. Exhaustively apply Rule 6 to (G, ϕ) and S_0, possibly changing ϕ.
2. If S_0 satisfies all demands of (G, ϕ) then return S_0 as a solution (and stop).
3. Otherwise, pick a vertex $v \in V(G)$ *of minimum nonzero demand*. Because we have exhaustively applied Rule 6 there must be a set C of less than $\phi(v) \leq d$ vertices that separates v from S_0 and all vertices of demand at least $\phi(v)$. Add $\{v\} \cup C$ to S_0 and increase ℓ by one. Note that we add at most $\phi(v) \leq d$ vertices to S_0 because $|C| < \phi(v)$.

After Step 3 the algorithm continues with Step 1. The following Invariant 8 guarantees that the algorithm runs for at most opt rounds. The approximation factor follows since only this step adds small "non-optimal" parts to the solution.

▶ **Invariant 8.** *There exists a set S_1 of at most $\mathsf{opt} - \ell$ vertices such that $S_0 \cup S_1$ is a feasible solution for (G, ϕ).*

▶ **Theorem 9.** *The* VECTOR d-CONNECTIVITY *problem admits a polynomial-time factor d-approximation.*

We can also derive an approximation algorithm for VECTOR CONNECTIVITY, where there is no fixed upper bound on the maximum demand. To this end, we can rerun the previous algorithm for all "guesses" of $\mathsf{opt}_0 \in \{1, \ldots, n\}$. In each run, we start with S_0 containing all vertices of demand greater than the guessed value opt_0, since those must be contained in every solution of total size at most opt_0. Then the maximum demand is $d = \mathsf{opt}_0$ and we get a d-approximate set of vertices to add to S_0 to get a feasible solution. When $\mathsf{opt}_0 = \mathsf{opt}$, then opt must also include the same set S_0 and for the remaining $\mathsf{opt} - |S_0| \leq \mathsf{opt}$ vertices we have a d-approximate extension; we get a solution of total size at most opt^2.

▶ **Corollary 10.** *The* VECTOR CONNECTIVITY *problem admits a polynomial-time approximation algorithm that returns a solution of size at most* opt^2*, where* opt *denotes the optimum solution size for the input.*

4 FPT algorithm for Vector Connectivity(k)

In this section we present a randomized FPT-algorithm for VECTOR CONNECTIVITY(k). (We recall that Lokshtanov announced this to be FPT.) Recall that the reduction rules in Section 2 also allow us to reduce the number of nonzero demand vertices to at most $k^3 + k$ (or safely reject). Based on this we are able to give a randomized algorithm building on a randomized FPT algorithm of Marx [8] for intersection of linear matroids. (The randomization comes from the need to construct a representation for the required matroids.) Concretely, this permits us to search for an independent set of size k in $k^3 + k$ linear matroids over the same ground set, where the independent set corresponds to the desired solution and each single matroid ensures that one demand vertex is satisfied.

▶ **Theorem 11.** VECTOR CONNECTIVITY(k) *is randomized fixed-parameter tractable without false positives and error probability exponentially small in the input size.*

Proof Sketch. W.l.o.g., input instances (G, ϕ, k) have at most k^3+k nonzero demand vertices (else apply the reduction rules); let $D = \{v \in V(G) \mid \phi(v) \geq 1\}$. Clearly, if the instance is YES then there exist also solutions of size *exactly* k (barring $|V(G)| < k$ which is trivial).

Algorithm. As a first step, we guess the intersection of a solution S^* of size k with the set D; there are at most $(k^3 + k)^k$ choices for $S_0 = D \cap S^*$. All vertices of demand exceeding k must be contained in S_0 for S^* to be a solution.

For each $v \in D \setminus S_0$, we construct a matroid M_v over $V' = V \setminus D$ as follows.

1. Build a graph G_v by first adding to G additional $c-1$ copies of v, called $v_2, \ldots, v_c$, where $c = \phi(v)$, and use $v_1 := v$ for convenience. Second, add $r = k - c$ universal vertices $w_1, \ldots, w_r$ (i.e., neighborhood $V \cup \{v_2, \ldots, v_c\}$).
2. Let M'_v denote the gammoid on G_v with source set $T = \{v_1, \ldots, v_c, w_1, \ldots, w_r\}$ and ground set $V' \cup S_0$. Recall that the independent sets of a gammoid are exactly those subsets I of the ground set that have $|I|$ vertex-disjoint paths from the sources to I. Gammoids are linear matroids and a representation over a sufficiently large field can be found in randomized polynomial time (cf. [8]).
3. Create M_v from M'_v by contracting S_0, making its ground set V'. If S_0 is independent in M'_v then any I is independent in M_v if and only if $S_0 \cup I$ is independent in M'_v.

Use Marx' algorithm [8] to search for a set I^* of size $k - |S_0|$ that is independent in each matroid M_v for $v \in D \setminus S_0$. If a set I^* is found then test whether $S_0 \cup I^*$ is a vector connectivity set for (G, ϕ, k) by appropriate polynomial-time flow computations. If yes then return the solution $S_0 \cup I^*$. Otherwise, if $S_0 \cup I^*$ is not a solution or if no set I^* was found then try the next set S_0, or answer NO if no set S_0 is left to check.

Correctness. Clearly, if (G, ϕ, k) is NO then the algorithm will always answer NO as all possible solutions $S_0 \cup I^*$ are tested for feasibility.

Assume now that (G, ϕ, k) is YES, let S^* a solution of size k, and let $S_0 = D \cap S^*$. Note that $S^* \subseteq V' \cup S_0$. Pick any $v \in D \setminus S_0$. It follows that there are $c = \phi(v)$ paths from v to S^* in G that are vertex-disjoint except for v. Thus, in G_v we get c (fully) vertex-disjoint paths from $\{v_1, \ldots, v_c\}$ to S^*, by giving each path a private copy of v. We get additional $r = k - c$ paths from $\{w_1, \ldots, w_r\}$ to the remaining vertices of S^* since $S^* \subseteq V' \cup S_0 \subseteq N(w_i)$. Thus, the set S^* is independent in each gammoid M'_v. Therefore, in each M'_v also $S_0 \subseteq S^*$ is independent. This implies that in M_v (obtained by contraction of S_0) the set $S^* \setminus S_0$ is independent and has size $k - |S_0|$. Moreover, any I is independent in M_v if and only if $I \cup S_0$ is independent in M'_v. It follows, from the former statement, that Marx' algorithm will find *some* set I of size $k - |S_0|$ that is independent in all matroids M_v for $v \in D \setminus S_0$.

We claim that $I \cup S_0$ is a vector connectivity set for (G, ϕ, k). Let $v \in D \setminus S_0$. We know that I is independent in M_v and, thus, $S := I \cup S_0$ is independent in M'_v. Thus, in G_v there are $|S| = k$ paths from T to S. This entails $c = \phi(v)$ vertex-disjoint paths from $\{v_1, \ldots, v_c\}$ to S that each contain no further vertex of T since $|T| = k$. By construction of G_v, we directly get $\phi(v)$ paths from v to S in G that are vertex-disjoint except for overlap in v. Thus, S satisfies the demand of any $v \in D \setminus S_0$. Since $S \supseteq S_0$, we see that S satisfies all demands. Thus, the algorithm returns a feasible solution, as required.

Runtime. Marx' algorithm for finding a set of size k' that is independent in ℓ matroids has FPT running time with respect to $k' + \ell$. We have $k' \leq k$ and $\ell \leq |D| \leq k^3 + k$ in all iterations of the algorithm and there are at most $(k^3 + k)^k$ iterations. This gives a total time that is FPT with respect to k, as claimed. ◀

5 Vertex-linear kernelization for constant demand

In this section we give an outline of our vertex-linear kernelization for VECTOR d-CONNECTIVITY(k), i.e., with d a constant and k the parameter. We will focus on explaining how we can directly bound the number of subproblems without using Bollobas' two-families theorem, whose bound depends on the size of reduced subproblems. Then we give some intuition about our explicit definition for equivalent subproblems and how to replace them.

The starting point for our kernelization are Reduction Rules 1 and 5, and a result of Cicalese et al. [3] that relates vector connectivity sets for (G, ϕ) to hitting sets for a family of connected subgraphs of G: The family contains all sets X such that $G[X]$ is connected and some vertex $v \in X$ has demand larger than $|N(X)|$; intuitively, every solution S must intersect each set X. We use instead a family $\mathcal{X}(G, \phi)$ containing only the *minimal* sets X. For ease of presentation we define $D(G, \phi) := \{v \in V(G) \mid \phi(v) \geq 1\}$, and use the shorthand $D = D(G, \phi)$ whenever G and ϕ are clear from context.

▶ **Proposition 12** (adapted from Cicalese et al. [3, Prop. 1]). *Let $G = (V, E)$, let $\phi \colon V \to \mathbb{N}$, and let $\mathcal{X} := \mathcal{X}(G, \phi)$. Then every set $S \subseteq V$ is a vector connectivity set for (G, ϕ) if and only if it is a hitting set for $\mathcal{X}$, i.e., it has a nonempty intersection with each $X \in \mathcal{X}$.*

For the general case of VECTOR CONNECTIVITY with unrestricted demands the size of $\mathcal{X}(G, \phi)$ can be exponential in $|V(G)|$; for VECTOR d-CONNECTIVITY there is a straightforward bound of $|\mathcal{X}| = \mathcal{O}(|V(G)|^d)$ since $|N(X)| \leq d - 1$. However, even for VECTOR d-CONNECTIVITY, the sets $X \in \mathcal{X}$ are not necessarily small and, thus, we will not take a hitting set approach for the kernelization; we will not materialize the set $\mathcal{X}$ but use it only for analysis.

To arrive at our kernelization we will later establish a reduction rule that shrinks connected subgraphs with small boundary and bounded number of demand vertices to constant size. This is akin to black-box protrusion-based reduction rules, especially as in [5], but we give an explicit algorithm that comes down to elementary two-way flow computations. To get an explicit (linear) bound for the number of subproblems, we introduce a new family $\mathcal{Y}$ with larger but (as we will see) fewer sets, and apply the reduction process to graphs $G[Y]$ with $Y \in \mathcal{Y}$. Alternatively, as pointed out in the introduction, one may use a result of Bollobas for bounding the number of sets in $\mathcal{X}$ once they are small; a caveat is that this bound would depend on the final size of sets in $\mathcal{X}$, whereas we have a direct and explicit bound for $|\mathcal{Y}|$.

▶ **Definition 13** ($\mathcal{Y}(G, \phi, d)$)**.** Let $G = (V, E)$, let $d \in \mathbb{N}$, and let $\phi \colon V \to \{0, \ldots, d\}$. The family $\mathcal{Y}(G, \phi, d)$ contains all sets $Y \subseteq V$ with

1. $G[Y]$ is connected,
2. $|Y \cap D| \leq d^3$, i.e., Y contains at most d^3 vertices v with nonzero demand,
3. $|N(Y)| \leq d$, i.e., Y has at most d neighbors, and
4. there is a vertex $v \in Y \cap D$, i.e., $\phi(v) \geq 1$, such that $N(Y)$ is the unique closest minimum $v, D \setminus Y$-separator.

We show that each set $X \in \mathcal{X}(G, \phi)$ is a subset of some $Y \in \mathcal{Y}(G, \phi, d)$.

▶ **Lemma 14.** *Let $G = (V, E)$, $d \in \mathbb{N}$, and $\phi \colon V \to \{0, \ldots, d\}$. Let $\mathcal{X} := \mathcal{X}(G, \phi)$ and $\mathcal{Y} := \mathcal{Y}(G, \phi, d)$. Then for all $X \in \mathcal{X}$ there exists $Y \in \mathcal{Y}$ with $X \subseteq Y$.*

We prove that the number of sets $Y \in \mathcal{Y}$ is linear in k for every fixed d, by analyzing a branching algorithm for finding $Y \in \mathcal{Y}$. Thus, by later shrinking the size of sets in $\mathcal{Y}$ to some constant we get $\mathcal{O}(k)$ vertices in total over sets $Y \in \mathcal{Y}$.

▶ **Lemma 15.** *Let (G, ϕ, k) an instance of* VECTOR d-CONNECTIVITY(k) *and let $\mathcal{Y} := \mathcal{Y}(G, \phi, d)$. Then $|\mathcal{Y}| \leq d^2 k \cdot 2^{d^3+d}$.*

5.1 Reducing the size of sets in $\mathcal{Y}$

Let us describe how to reduce the size of sets $Y \in \mathcal{Y}$ through modifications on the graph G. At a high level, this will be achieved by replacing subgraphs $G[Y]$ by "equivalent" subgraphs of bounded size. When this is done, we know that the total number of vertices in sets $Y \in \mathcal{Y}$ is $\mathcal{O}(k)$. Since this part is somewhat technical, the proof details are deferred to a full version.

Consider a set $Y \in \mathcal{Y}$ and its (small) neighborhood $Z := N_G(Y)$. Think of deciding whether (G, ϕ, k) is YES as a game between two players, Alice and Bob. Alice sees only $G[Y \cup Z]$ and wants to satisfy the demands of all vertices in Y, and Bob sees only $G - Y$ and wants to satisfy the demands of the vertices in $V \setminus Y$. To achieve a small solution the players must cooperate and exchange information about paths between vertices in Z, or between Z and vertices of a partial solution, that they can *provide* or that they *require* from the other player.

Crucially, we know that there is only a constant number of nonzero demand vertices in Y, which can be seen to imply that the intersection of optimal solutions with Y is bounded. Thus, Alice can try all partial solutions $S_Y \subseteq Y$ of bounded size and determine what paths between Z and S_Y or between different vertices of Z she can provide for Bob. We formalize this set of paths as her *facilities.* She can furthermore determine what paths she needs to satisfy the demand of each of her vertices. Each demand vertex v gives rise to a set of required paths that may run between Z and Bob's solution or just between vertices of Z. We formalize this family of sets as Alice's *requirement.* (Note that in fact both players can offer or require various different *packings* of such paths.)

We now define Alice's requirements formally; the facilities are deferred to a full version. For notational convenience, we call a set of paths to be v*-independent* if each pair of paths is vertex-disjoint except for possibly sharing v as an endpoint. For a graph G, a vertex v, an integer i, and two vertex subsets $A, B \subseteq V(G)$ we define a (v, i, A, B)-*constrained path packing* as a set of $i + |A|$ v-independent paths from $A \cup \{v\}$ to B in G. Herein we explicitly allow $v \notin V(G)$ if $i = 0$. We tacitly assume that, if $A \cap B \neq \emptyset$ then the paths corresponding to $A \cap B$ in the packing are of length zero, that is, they each comprise a single vertex.

Satisfying connectors indicate which of Bob's path packings are sufficient for Alice and a certain demand vertex.

▶ **Definition 16** (Satisfying connector). Let H be a graph on vertex set $Y \uplus Z$, let $v \in Y$, let v have positive demand d_v, and let $S_Y \subseteq Y$ be a partial solution. A tuple (A, B, C) with $A, B, C \subseteq Z$, pairwise disjoint, is a *satisfying connector for v with respect to S_Y in H* if either $v \in S_Y$ or there is a $(v, d_v, A, B \cup S_Y)$-constrained path packing in $H - C$. The set of all satisfying connectors for v with respect to the partial solution S_Y is denoted by $\mathit{Sat}(H, Z, d_v, S_Y, v)$.

Intuitively, if Bob can send disjoint paths from B to his part of the solution and to A, possibly using vertices of C, then Alice can complete these paths to satisfy the demand of v. The requirement now formalizes the notion that each partial solution for Alice gives rise to a certain set of satisfying connectors.

▶ **Definition 17** (Requirement). Let H be a graph on vertex set $Y \uplus Z$, let ϕ be a demand function on Y, and let $S_Y \subseteq Y$ be a partial solution. The *requirement* $\mathit{Req}(H, Z, \phi, S_Y)$ is the collection $\{\mathit{Sat}(H, Z, \phi(v), S_Y, v) \mid v \in D(H, \phi) \cap Y\}$.

It is not hard to observe that a partial solution S_Y in some $Y \in \mathcal{Y}$ has size at most $d^3 + d$ without loss of generality (recall that $|Y \cap D| \leq d^3$). Based on this fact, we can prove that $G[Y \cup Z]$ can be safely replaced by any graph G' that has the same set of facilities and the same family of requirements as $G[Y \cup Z]$ for any choice of S_Y with $|S_Y| \leq d^3 + d$. A smallest such graph G' has size bounded by some function of $|Z|$ since, if $S_Y \leq d^3 + d$, then the family of requirements has size bounded by some function in d. Checking whether the families of requirements of $G[Y \cup Z]$ and a candidate G' match can be done using a series of maximum flow computations that exploit the definition of satisfying connectors. This means that it is feasible to search incrementally for a representative G' of smallest (constant) size with the same requirement as $G[Y \cup Z]$. Similarly, we can ensure that the facilities are the same.

5.2 Kernelization procedure

Given an instance (G, ϕ, k) of VECTOR d-CONNECTIVITY(k) the kernelization proceeds as follows. Throughout, we refer to the current instance by (G, ϕ, k) and recall the use of $D = \{v \in V(G) \mid \phi(v) \geq 1\}$.

1. Apply Rule 1 exhaustively and then apply Rule 5 (this may return answer NO if we have more than d^2k demand vertices).
2. Apply the above described reduction rule once that may replace a subgraph $G[Y]$ by a smaller, constant-size graph.
3. Return to Step 1 if Step 2 was successful.
4. Let $W := D \cup \bigcup_{Y \in \mathcal{Y}} N[Y]$. Perform the torso operation on W in G to obtain G'. That is, carry out the following steps:
   a. Start with $G' = G[W]$.
   b. For every pair $u, v \in W$, if there is a u, v-path in G with internal vertices from $V \setminus W$ then add the edge $\{u, v\}$ to G'.
5. Return (G', ϕ', k) as the kernelized instance, where ϕ' is ϕ restricted to W.

Intuitively, the torso operation in Step 4 removes all vertices that are not in any set $Y \in \mathcal{Y}$ without affecting the sets therein. Overall, we obtain the following.

▶ **Theorem 18.** VECTOR d-CONNECTIVITY(k) *has a vertex-linear kernelization.*

6 Kernelization lower bound

In this section, we prove that VECTOR CONNECTIVITY(k) admits no polynomial kernelization unless NP $\subseteq$ coNP/poly. We give a reduction from HITTING SET(m), i.e., HITTING SET with parameter number of sets, which also makes a polynomial Turing kernelization unlikely (cf. [6]). Since demands greater than $k+1$ can be safely replaced by demand $k+1$, implying $d \le k+1$, the lower bound applies also to parameterization by $d+k$.

▶ **Theorem 19.** VECTOR CONNECTIVITY(k) *does not admit a polynomial kernelization unless* NP $\subseteq$ coNP/poly *and the polynomial hierarchy collapses.*

Proof. We give a polynomial parameter transformation from HITTING SET(m) to VECTOR CONNECTIVITY(k), which is known to imply the claimed lower bound (cf. [1]). Let $(U, \mathcal{F}, k)$ be an instance of HITTING SET(m) with parameter $m = |\mathcal{F}|$; w.l.o.g. $k \le m$. Let $n := |U|$. We construct a graph G on $2(k+1)m + n$ vertices that has a vector connectivity set of size at most $k' = (k+1)m + k = \mathcal{O}(m^2)$ if and only if $(U, \mathcal{F}, k)$ is YES for HITTING SET(m).

Construction. Make one vertex x_u for each element $u \in U$, and make $2(k+1)$ vertices $y_{1,F}, \ldots, y_{k+1,F}, y'_{1,F}, \ldots, y'_{k+1,F}$ for each set $F \in \mathcal{F}$. We add the following edges:

1. Add $\{y_{i,F}, y'_{i,F}\}$ for all $i \in \{1, \ldots, k+1\}$ and $F \in \mathcal{F}$.
2. Add $\{x_u, y_{i,F}\}$ for all $i \in \{1, \ldots, k+1\}$, $F \in \mathcal{F}$, and $u \in F$.
3. Make the set of all vertices $y_{i,F}$ a clique (not including any y'-vertex).

Set the demand ϕ of each $y'_{i,F}$ vertex to 2 and of each $y_{i,F}$ vertex to $(k+1)m+1$; all x-vertices have demand zero. Set the budget k' to $(k+1)m + k$. This completes the construction of an instance (G, ϕ, k'), which can be easily performed in polynomial time.

Correctness. Assume first that (G, ϕ, k') is YES and let S a vector connectivity set of size at most k'. Note that S must contain all vertices $y'_{i,F}$ since they have demand of 2 but only one neighbor (namely $y_{i,F}$). This accounts for $(k+1)m$ vertices in S; there are at most k further vertices in S. Let T contain exactly those elements $u \in U$ such that $x_u \in S$; thus $|T| \le k$. We claim that T is a hitting set for $\mathcal{F}$. Let $F \in \mathcal{F}$ and assume that $T \cap F = \emptyset$. It follows that S contains no vertex x_u with $u \in F$. Since at most k vertices in S are not y'-vertices, we can choose $i \in \{1, \ldots, k+1\}$ such that S does not contain $y_{i,F}$. Consider the set C consisting of all y-vertices other than $y_{i,F}$ as well as the vertex $y'_{i,F}$. In $G - C$ we find a connected component containing $y_{i,F}$ and all x_u with $u \in F$ but no further vertices. Crucially, all other neighbors of $y_{i,F}$ are $y'_{i,F}$ and all y-vertices, and x-vertices only have y-vertices as neighbors. By assumption S contains no vertex of this connected component. This yields a contradiction cause C is of size $(k+1)m$ and separates $y_{i,F}$ from S, but since S is a solution with $y_{i,F} \notin S$ there should be $(k+1)m+1$ disjoint paths from $y_{i,F}$ to S. Thus, S must contain some x_u with $u \in F$, and then $T \cap F \neq \emptyset$.

Now, assume that $(U, \mathcal{F}, k)$ is YES for HITTING SET(m) and let T a hitting set of size at most k for $\mathcal{F}$. We create a vector connectivity set S by selecting all x_u with $u \in T$ as well as all y'-vertices; thus $|S| \le k' = (k+1)m + k$. Clearly, this satisfies all y'-vertices. Consider any vertex $y_{i,F}$ and recall that its demand is $\phi(y_{i,F}) = (k+1)m+1$. We know that S contains at least one vertex x_u with $u \in F$ that is adjacent to $y_{i,F}$. Thus, we can find the required $(k+1)m+1$ disjoint paths from $y_{i,F}$ to S:

- We have one path $(y_{i,F}, y'_{i,F})$ and one path $(y_{i,F}, x_u)$.
- For all $(j, F') \neq (i, F)$ we get one path $(y_{i,F}, y_{j,F}, y'_{j,F})$; we get $(k+1)m - 1$ paths total.

It follows that (G, ϕ, k') is YES for VECTOR CONNECTIVITY(k).

We have given a polynomial parameter transformation from HITTING SET(m), which is known not to admit a polynomial kernelization unless $\mathsf{NP} \subseteq \mathsf{coNP/poly}$ [4] (see also [6]). This is known to imply the same lower bound for VECTOR CONNECTIVITY(k) [1]. ◀

7 Conclusion

We have presented kernelization and approximation results for VECTOR CONNECTIVITY and VECTOR d-CONNECTIVITY. An important ingredient of our results is a reduction rule that reduces the number of vertices with nonzero demand to at most $d^2\mathsf{opt}$ (or, similarly, to at most $\mathsf{opt}^3 + \mathsf{opt}$ or $k^3 + k$). From this, one directly gets approximation algorithms with ratios d^2 and $(\mathsf{opt}^2 + 1)$; we improved these to factors d and opt, respectively, by a local-ratio type algorithm. Recall that VECTOR d-CONNECTIVITY is APX-hard already for $d = 4$ [3].

On the kernelization side we show that VECTOR CONNECTIVITY(k) does not admit a polynomial kernelization unless $\mathsf{NP} \subseteq \mathsf{coNP/poly}$. Since demands greater than $k + 1$ can be safely replaced by demand $k + 1$ (because they cannot be fulfilled without putting the vertex into the solution) the lower bound extends also to parameter $k + d$. For VECTOR d-CONNECTIVITY(k), where d is a problem-specific constant, we give an explicit vertex-linear kernelization with at most $f(d) \cdot k = \mathcal{O}(k)$ vertices; the computable function $f(d)$ is superpolynomial in d, which is necessary (unless $\mathsf{NP} \subseteq \mathsf{coNP/poly}$) due to the lower bound for $d + k$.

Finally, the reduction to $\ell \leq k^3 + k$ nonzero demand vertices allows an alternative proof for fixed-parameter tractability: We give a randomized FPT-algorithm for VECTOR CONNECTIVITY(k) that finds a solution by seeking a set of size k that is simultaneously independent in each of ℓ linear matroids; for this we use an algorithm of Marx [8] for linear matroid intersection, which is fixed-parameter tractable in $k + \ell = \mathcal{O}(k^3)$.

Acknowledgments. The authors are grateful to anonymous reviewers for suggesting a simplified definition for signatures and pointing out the non-constructive kernelization argument that can be obtained from the work of Fomin et al. [5].

References

1 Hans L. Bodlaender, Bart M. P. Jansen, and Stefan Kratsch. Kernelization lower bounds by cross-composition. *SIAM J. Discrete Math.*, 28(1):277–305, 2014.

2 Endre Boros, Pinar Heggernes, Pim van 't Hof, and Martin Milanič. Vector connectivity in graphs. *Networks*, 63(4):277–285, 2014.

3 Ferdinando Cicalese, Martin Milanič, and Romeo Rizzi. On the complexity of the vector connectivity problem. *Theor. Comput. Sci.*, 591:60–71, 2015.

4 Michael Dom, Daniel Lokshtanov, and Saket Saurabh. Kernelization lower bounds through colors and IDs. *ACM T. Alg.*, 11(2):13:1–13:20, 2014.

5 Fedor V. Fomin, Daniel Lokshtanov, Saket Saurabh, and Dimitrios M. Thilikos. Linear kernels for (connected) dominating set on graphs with excluded topological subgraphs. In *Proc. 30th STACS*, pages 92–103. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2013.

6 Danny Hermelin, Stefan Kratsch, Karolina Soltys, Magnus Wahlström, and Xi Wu. A completeness theory for polynomial (Turing) kernelization. *Algorithmica*, 71(3):702–730, 2015.

7 Stasys Jukna. *Extremal combinatorics - with applications in computer science.* Texts in theoretical computer science. Springer, 2001.

8 Dàniel Marx. A parameterized view on matroid optimization problems. *Theor. Comput. Sci.*, 410(44):4471–4479, 2009.