A Parameterized Algorithm for Vertex Connectivity Survivable Network Design Problem with Uniform Demands

Jørgen Bang-Jensen
University of Southern Denmark, Odense, Denmark

Kristine Vitting Klinkby
University of Southern Denmark, Odense, Denmark

Pranabendu Misra
Chennai Mathematical Institute, India

Saket Saurabh
Institute of Mathematical Sciences, Chennai, India
University of Bergen, Norway

Abstract

In the Vertex Connectivity Survivable Network Design (VC-SNDP) problem, the input is a graph G and a function $d : V(G) \times V(G) \to \mathbb{N}$ that encodes the vertex-connectivity demands between pairs of vertices. The objective is to find the smallest subgraph H of G that satisfies all these demands. It is a well-studied NP-complete problem that generalizes several network design problems. We consider the case of *uniform demands*, where for every vertex pair (u, v) the connectivity demand $d(u, v)$ is a fixed integer κ. It is an important problem with wide applications.

We study this problem in the realm of Parameterized Complexity. In this setting, in addition to G and d we are given an integer ℓ as the *parameter* and the objective is to determine if we can remove at least ℓ edges from G without violating any connectivity constraints. This was posed as an open problem by Bang-Jansen et.al. [SODA 2018], who studied the edge-connectivity variant of the problem under the same settings. Using a powerful classification result of Lokshtanov et al. [ICALP 2018], Gutin et al. [JCSS 2019] recently showed that this problem admits a (non-uniform) FPT algorithm where the running time was unspecified. Further they also gave an (uniform) FPT algorithm for the case of $\kappa = 2$. In this paper we present a (uniform) FPT algorithm any κ that runs in time $2^{O(\kappa^2 \ell^4 \log \ell)} \cdot |V(G)|^{O(1)}$.

Our algorithm is built upon new insights on vertex connectivity in graphs. Our main conceptual contribution is a novel graph decomposition called the *Wheel decomposition*. Informally, it is a partition of the edge set of a graph G, $E(G) = X_1 \cup X_2 \ldots \cup X_r$, with the parts arranged in a cyclic order, such that each vertex $v \in V(G)$ either has edges in at most two consecutive parts, or has edges in every part of this partition. The first kind of vertices can be thought of as the rim of the wheel, while the second kind form the hub. Additionally, the vertex cuts induced by these edge-sets in G have highly symmetric properties. Our main technical result, informally speaking, establishes that "nearly edge-minimal" κ-vertex connected graphs admit a wheel decomposition – a fact that can be exploited for designing algorithms. We believe that this decomposition is of independent interest and it could be a useful tool in resolving other open problems.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms

Keywords and phrases Parameterized Complexity, Vertex Connectivity, Network Design

Digital Object Identifier 10.4230/LIPIcs.ESA.2023.13

Funding *Pranabendu Misra*: Supported by Google India Research Award 2022, and Start-Up Grant 2022 (SRG/2022/001927) of Science and Engineering Research Board (SERB), India.
Saket Saurabh: Supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 819416), and Swarnajayanti Fellowship (No. DST/SJF/MSA01/2017-18).

1 Introduction

One of the most important challenges in designing real world networks is to ensure their reliability in the face of damages and equipment failures. A natural solution is to build additional redundancy in the network, at lowest possible costs, to guarantee connectivity up to a certain number of failures. This motivates the study of *network design problems*, and they are well studied in combinatorial optimization and algorithm design. The problem is abstractly described using graphs, where vertices naturally represent the nodes of the network, the edges represent the connections between the nodes and each edge has an associated cost. In many applications, the objective is to find a minimum cost subgraph that connects all the nodes and is also able to withstand a certain number of failures of vertices or edges. We refer to the surveys [14, 19, 22] for details.

Formally, in the case of vertex failures this problem is called Minimum κ-Vertex Connected Spanning Subgraph (κ-VCSS); the input is a graph G on n vertices with edge costs $w : E(G) \to \mathbb{R}^+$, and the objective is to find a spanning subgraph H of minimum total cost such that it remains connected even when $\kappa - 1$ vertices are deleted, for some fixed integer κ. And in the case of edge-failures, we get Minimum λ-Edge Connected Spanning Subgraph (λ-ECSS) where we must ensure network connectivity even after $\lambda - 1$ edges fail, for some fixed integer λ. Both these problems are very well studied, and have wide applications. Unfortunately they are NP-hard, even in the unweighted setting. Therefore they have been extensively studied in Approximation algorithms [14, 19, 22], and more recently in Parameterized algorithms.

If a pair of vertices $u, v \in V(G)$ remain connected even after $\kappa - 1$ vertex failures, then by the Menger's Theorem [3, Theorem 7.3.1], there must be κ internally disjoint paths from u to v in G. In other words, the *vertex-connectivity between u and v* is at least κ in G. Since every pair of vertices in G must have this property, the graph G must be κ-vertex connected. Similarly, in the case of edge-connectivity, the graph G must be λ-edge connected. Interpreting κ-VCSS / λ-ECSS in terms of connectivity leads to the Survivable Network Design Problem (SNDP), where we may have different connectivity demands between different vertex pairs. SNDP captures a number of network design problems such as Steiner Tree, Minimum Equivalent Digraph, Hamiltonian Cycle etc. We can classify many network design problems by the nature of their connectivity constraints into variants of Edge Connectivity SNDP (EC-SNDP) and Vertex Connectivity SNDP (VC-SNDP). Observe that κ-VCSS and λ-ECSS correspond to the uniform connectivity demands case of these problems, respectively. Hence they are also called VC-SNDP / EC-SNDP with Uniform Demands. There has been a vast amount on research network design problems, especially in Approximation algorithms [14, 19, 22], which has led to the development of a number of new algorithmic techniques; e.g. the 2-approximation algorithm of Jain for EC-SNDP [13] which introduced iterative LP-rounding.

Typically, vertex connectivity problems are often significantly more difficult than the corresponding edge-connectivity problems. In contrast to the 2-approximation for EC-SNDP, VC-SNDP has an approximation lower bound of $2^{\log^{1-\epsilon} n}$ [18], and κ^ϵ for every $\kappa > \kappa_0$ where $\epsilon > 0$ and $\kappa_0 > 1$ are constants [6]. Even in the case of uniform requirements, i.e. κ-VCSS, the best known approximation algorithm for κ-VCSS has an approximation factor $\mathcal{O}(\log \kappa \cdot \log \frac{n}{n-\kappa})$[23], although it is improved to 6 assuming that $n \geq \kappa^3(\kappa - 1) + \kappa$ [7]. Another example is the Unrestricted Vertex Connectivity Augmentation (UVCA), where the objective is to augment a κ-vertex connected graph to a $\kappa + t$-vertex connected

graph by adding new edges.[1] While Unrestricted Edge Connectivity Augmentation (UECA), that is similarly defined, is known to be in polynomial time from many decades ago[11], the complexity of UVCA still remains open. Currently, a polynomial time algorithm is only known for $t = 1$ [26].

Recently, there has been a lot of interest in the study of the Parameterized algorithms [8][2] for network design problems [2, 21, 4, 1, 12, 15, 16, 17, 10, 9]. The separation between vertex-connectivity problems and edge-connectivity problems also persists in this setting; nearly all of the currently known results are for edge-connectivity problems. Bang-Jensen et al. [1] gave an FPT algorithm for the parameterized version of λ-ECSS[3] in this setting. This naturally raises the question for κ-VCSS, denoted by p-κ-VCSS[4], which they pose as an open problem. This is the problem we study here. Gutin et.al. [12] gave a *non-uniform FPT algorithm*[5] for p-κ-VCSS using a powerful classification result of Lokshtanov et al. [20]. The running time of this algorithm is unspecified. They also gave an (uniform) FPT algorithm for p-κ-VCSS when $\kappa = 2$ (Biconnectivity Deletion) [12].

p-κ-VCSS **Parameter:** ℓ
Input: A κ-connected graph $G = (V, E)$ and an integer ℓ.
Question: Does there exist a set of edges $F \subseteq E$ such that $|F| \geq \ell$ and $G' = (V, E \setminus F)$ is κ-connected?

Our contribution. In this work we give an (uniform) FPT algorithm for p-κ-VCSS for any constant κ; indeed our result is stronger and the algorithm is FPT in both ℓ and κ. We build upon the broad approach of Bang-Jensen et al. [1], but develop new insights into vertex connectivity that are of independent interest.

▶ **Theorem 1.** p-κ-*VCSS admits an FPT algorithm running in time* $2^{\mathcal{O}(\kappa^2 \ell^4 \log \ell)} \cdot |G|^{\mathcal{O}(1)}$.

Our algorithm follows the general scheme laid down by Bang-Jensen et al. [1] for p-λ-ECSS. We consider the set of all *deletable edges*, i.e. all those edges $e \in G(G)$ such that $G - e$ is κ-vertex connected. Our goal is to identify an *irrelevant edge*[6] among these edges and shrink the pool of *relevant* deletable edges until the instance can be more easily solved. Bang-Jensen et al. [1] show that, for p-λ-ECSS, if the graph contains a large number of deletable edges, then there exists a cyclic decomposition of the graph that can be used to identify an irrelevant edge.[7] We prove a similar kind of result for p-κ-VCSS.

We develop a new tool called the *Wheel decomposition* for this purpose that helps us understand the structure of "nearly edge minimal" κ-vertex connected graphs. This is our main conceptual contribution. An intuitive description is as follows: It is a partition of the

[1] Formally, the input is a κ-vertex connected graph G and an integer $t \geq 1$. The objective is to compute a minimum cost subset $F \subseteq V \times V$ such that $G + F$ is $(\kappa + t)$-vertex connected.
[2] Let us recall that in Parameterized Complexity, the input is a pair (X, ℓ) where X is an instance of the problem and ℓ is an integer, called the *parameter*. The objective is to design a *Fixed Parameter Tractable (FPT) algorithm*, i.e. an algorithm that solves the problem in time $f(\ell) \cdot |X|^{\mathcal{O}(1)}$ where f is a function of ℓ alone.
[3] They give FPT algorithms for λ-ECSS in both graphs and digraphs. Further, they extended their results to κ-VCSS in digraphs by reducing it to special instances of λ-ECSS.
[4] Here the p in p-κ-VCSS denotes *parameterized*
[5] Here *non-uniform algorithm* refers to the complexity theory term, which is different from demands being uniform or non-uniform.
[6] i.e. one that is not present in some solution to the input instance and therefore remains in the graph when the edges of that solution are deleted.
[7] This is specifically required for odd values of λ in undirected graphs. The other cases are much simpler.

edge set $E(G)$ of a graph G, with the parts arranged in a cyclic order. Any vertex either has edges in at most two consecutive parts, or else it has edges in every part of the partition.[8] Each of these parts, which are edge-subsets, describes a vertex cut in a natural way: the set of all those vertices that have an edge in this part and another edge somewhere outside it. In a Wheel decomposition, these vertex cuts are highly symmetric. For our results, we develop algorithmic tools to construct Wheel decompositions in instances of p-κ-VCSS and then identify irrelevant edges using them.

We believe that the Wheel decomposition is of independent interest. As we have observed earlier, vertex connectivity problems are often substantially more difficult than the corresponding edge-connectivity problems. One reason for this is the lack of structural results and tools for vertex connectivity as compared to edge connectivity. We believe that Wheel decompositions will be a useful tool in understanding vertex connectivity and resolving other open questions in the future.

Related works. As we mention above, our work has been most influenced by the work of Bang-Jensen et al. [1]; we however require several new ideas and methods to deal with vertex connectivity. Other related works gave FPT algorithms for Directed and Undirected Spanners [16, 17, 10], and 2-VCSS [12] in a similar setting. A recent work considered these problems parameterized by the solution size [9]. Additional results are known about connectivity augmentation problems, e.g. Strong Connectivity Augmentation [15], Edge connectivity Augmentation by One [21, 4] and Minimum Strong Spanning Subgraph [2].There is a vast amount of research on network design problems in approximation algorithms, too many to list comprehensively. We refer to the following surveys for an overview [14, 19, 22]. There has been recent interest in improving the approximation-factor for some specific simple cases of EC-SNDP, such as Weighted Tree Augmentation and Weighted Connectivity Augmentation by One). A sequence of works have finally led to breaching the approximation factor barrier 2 for both these problems [5, 24, 25]. Ideas and methods from the Parameterized algorithms for these problems also played a role in some of these results [5, 4].

2 Preliminaries and Notation

For a universe W and a subset of elements $U \subseteq W$ we will denote the set $W \setminus U$ by $\overline{U}$. When doing set-operations where one of the sets is a singleton, e.g. $A \cup \{x\}$, we omit the curly-braces and just write $A \cup x$. For a graph G and a subset of vertices $A \subseteq V(G)$, $G - A$ denotes the induced subgraph $G[V(G) \setminus A]$. Similarly, for a set of edges $B \subseteq E(G)$, $G - B$ denotes the subgraph with vertex set $V(G)$ and edge-set $E(G) \setminus B$. For a graph G and a subset of vertex-pairs $B \subseteq V \times V$, $G + B$ denotes the graph with vertex set $V(G)$ and edge-set $E(G) \cup B$. Let $G = (V, E)$ be a graph and for an edge $e = uv \in E$ the **endpoints of e** are the vertices u and v, and we have $V(e) = \{u, v\}$. For a vertex $u \in V$ let $\delta(u) \in E$ define the set of edges which have u as an endpoint. We extend this notion to subsets of vertices, i.e. $\delta(X)$ denotes the set of edges with an endpoint in X. Further, for an edge $e = (u, v)$, $\delta(e)$ denotes the set $\delta(V(e))$. For a set of edges $E' \subset E$ and a vertex v we will often denote that $\delta(v) \subseteq E'$ by writing $v \in E'$. Given a graph $G = (V, E)$, two vertices $s, t \in V$, and a path P from s to t we denote the vertices of P as $V(P)$ and $E(P)$ will denote the edges of P. Given two vertices $u, v \in V(P)$ the path $P[u, v]$ is the subpath of P which goes from u to v. Furthermore, $P]u, v[$ will be the induced path $P[u, v] \setminus \{u, v\}$.

[8] These two types of vertices can be thought of as the rim and the hub of a wheel.

An instance of a parameterized problem Π consists of a main part I and a parameter l, and it is denoted denoted (I, l). A parameterized decision problem Π admits a **fixed-parameter tractable (FPT)** if for every instance $(I, \ell) \in \Pi$ the problem admits an algorithm which can decide if there is an affirmative or negative answer to the problem and the algorithm has complexity $O(f(\ell)|I|^c)$ for some positive function f. The problem admits a *uniform FPT algorithm*, if there is an algorithm $\mathcal{A}$ that solves an instance (I, ℓ) of the problem in $O(f(\ell)|I|^c)$ time. And it admits a *non-uniform FPT algorithm* if for each value of ℓ, there is an algorithm $\mathcal{A}_\ell$ that solves an instance (I, ℓ) in $O(f(l)|I|^c)$ time.

Vertex-cuts and Separators

▶ **Definition 2.** *Given a set of edges $U \subseteq E$ the **vertex-cut** induced by U is the minimal set of vertices denoted $\mathcal{S}(U) \subseteq V$ such that for every pair of edges $e \in U$ and $e' \in \overline{U}$ we have $V(e) \cap V(e') \subseteq \mathcal{S}(U)$.*

Note that, for every vertex $u \in \mathcal{S}(U)$ the sets $\delta(u) \cap U$ and $\delta(u) \cap \overline{U}$ are both not empty due to minimality of U. For two disjoint sets of edges $U, W \subseteq E$ we will use the notation $\mathcal{D}(U, W)$ to define those vertices in V which are endpoints of edges in both U and W. For two different graphs $G = (V, E)$ and $G' = (V, E')$, defined on the same set of vertices V, and a set of edges U such that $U \subseteq E \cap E'$ we will use a subscript to indicate in which graph we consider the vertex-cut $\mathcal{S}(U)$. That is, $\mathcal{S}(U)_G$ will refer to a (uniquely determined) vertex-cut in G while $\mathcal{S}(U)_{G'}$ will refer to a (uniquely determined) vertex-cut in G'. Moreover, with a slight abuse of notation for the sake of brevity, for a set of edges $W \subseteq E$ and the graph $G^* = (V, E \setminus W)$ we denote the vertex-cut $\mathcal{S}(U \setminus W)_{G^*}$ in G^* as $\mathcal{S}(U)_{G^*}$. Note that, the underlying graph of a vertex-cut and the corresponding subset of edges are always clear from the context. Note that $\mathcal{S}(U)_{G^*} \subseteq \mathcal{S}(U)_G$. We will omit the subscript if there is only one graph or it is explicitly given in which graph the vertex-cut should be considered. Similarly, for sets of edges and sets of vertices a subscript will indicate which graph we refer to.

A set of vertices $Q \subseteq V$ in a graph $G = (V, E)$ is a **vertex-separator** if there exist two vertices $u, v \in V \setminus Q$ such that there is no path from u to v in $G - Q$. We will also call such a vertex-separator Q a **(u, v)-separator** and say that Q **separates** u and v. For a minimal (u, v)-separator Q, let R be the set of vertices in the same connected component as u in $G - Q$ and $T = V \setminus (R \cup Q)$. Now let $U \subseteq E$ be the set of edges which have an endpoint in R and observe that $Q = \mathcal{S}(U)$, that is, the vertex-cut $\mathcal{S}(U)$ is a minimal (u, v)-separator. This implies that $\delta(u) \subseteq U$ and $\delta(v) \subseteq \overline{U}$. Conversely, if there exists a vertex-cut $\mathcal{S}(U)$ such that for two vertices $u, v \in V$ it holds that $\delta(u) \subseteq U$ and $\delta(v) \in \overline{U}$ then there cannot be a path from u to v in $G - \mathcal{S}(U)$ as $\delta(v)$ will be disconnected from $\delta(u)$ in this graph. Hence $\mathcal{S}(U)$ must be a (u, v)-separator. Therefore, we have the following observation:

▶ **Observation 3.** *Let $G = (V, E)$ be a graph. For $U \subseteq E$, if there exist two vertices $u, v \in V$ such that $\delta(u) \subseteq U$ and $\delta(v) \in \overline{U}$, then $\mathcal{S}(U)$ is a (u, v)-vertex-separator. And if Q is a minimal (u, v)-separator, then there is a subset of edges $U \subseteq E$ such that $Q = \mathcal{S}(U)$, $\delta(u) \subseteq U$ and $\delta(v) \subseteq \overline{U}$.*

We call a vertex-cut $\mathcal{S}(U)$ a **nearby vertex-separator with respect to** $e = uv$ if exacty one of the vertices u and v is contained in $\mathcal{S}(U)_G$, and $\mathcal{S}(U)_{G-e}$[9] is a (u, v)-separator in $G - e$. Figure 1 shows a nearby vertex-separator. By Definition 2, for every $x \in \mathcal{S}(U)_{G-e}$ it holds that $\delta(x)_{G-e}$ intersects U_{G-e} and $\overline{U}_{G-e}$. Hence $\mathcal{S}(U)_G = \mathcal{S}(U)_{G-e} \cup v$ for $v \in V(e)$ which implies that $|\mathcal{S}(U)_G| = |\mathcal{S}(U)_{G-e}| + 1$.

[9] Recall that, for brevity we write $\mathcal{S}(U)_{G-e}$ to denote $\mathcal{S}(U \setminus \{e\})_{G-e}$.

▶ **Observation 4.** *For a graph $G = (V, E)$ and a nearby vertex-separator $\mathcal{S}(U)_G$ with respect to $e = uv$ it holds that $|\mathcal{S}(U)_G| = |\mathcal{S}(U)_{G-e}| + 1$ and $\mathcal{S}(U)_G = \mathcal{S}(U)_{G-e} \cup v$ for some $v \in V(e)$.*

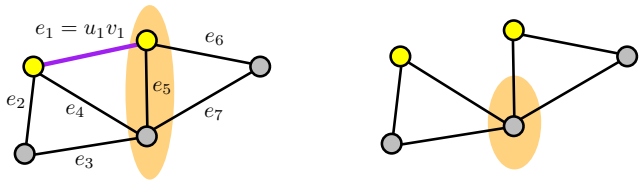

Figure 1 $\mathcal{S}(U)$ is a near-vertex separator for e_1 where $U = \{e_1, e_2, e_3, e_4\}$.

▶ **Observation 5.** *A vertex-cut $\mathcal{S}(U)$ is a nearby vertex-separator with respect to $e = uv$ if and only if one of the following holds:*

- *$\delta(u) \setminus e \subseteq U$ and $\delta(v) \subseteq \overline{U}$ (or $\delta(u) \setminus e \subseteq \overline{U}$ and $\delta(v) \subseteq U$)*
- *$\delta(u) \subseteq U$ and $\delta(v) \setminus e \subseteq \overline{U}$ (or $\delta(u) \subseteq \overline{U}$ and $\delta(v) \setminus e \subseteq U$)*

We have the following results which follows directly from Menger's Theorem.

▶ **Lemma 6.** *Given a graph $G = (V, E)$ and two vertices $u, v \in V$ such that $uv \notin E$, it is possible to find a minimum (u, v)-separator $\mathcal{S}(U)$ in $O((|V| + |E|)^{O(1)})$ time.*

▶ **Corollary 7.** *Given a graph $G = (V, E)$ and an edge e, in $O((|V| + |E|)^{O(1)})$ time it is possible to find a minimum nearby vertex-separator with respect to e or determine that no nearby vertex-separator with respect to e exists.*

▶ **Theorem 8.** *The following statements are equivalent:*

- *The graph $G = (V, E)$ is κ-connected.*
- *$|V| \geq \kappa + 1$ and for every pair of vertices $u, v \in V$ such that $uv \notin E$ there are κ internally disjoint paths from u to v.*
- *$|V| \geq \kappa + 1$ and every vertex-separator has size at least κ.*

▶ **Lemma 9.** *Given a graph $G = (V, E)$, it is possible to determine if G is κ-connected in $O(|G|^{O(1)})$ time .*

For vertex-cuts in general we have the following.

▶ **Lemma 10.** *The size of vertex-cuts is a submodular function, that is, given two edge-cuts $\mathcal{S}(U)$, $\mathcal{S}(W)$ we have $|\mathcal{S}(U \cap W)| + |\mathcal{S}(U \cup W)| \leq |\mathcal{S}(U)| + |\mathcal{S}(W)|$.*

Deletable and Relevant Edges

For a p-κ-VCSS instance $(G = (V, E), \ell)$ where G is a κ-vertex connected graph, we call a subset of edge $F \subseteq E$ **a solution** if $G = (V, E \setminus F)$ is κ-connected and $|F| \geq l$. To help distinguish between the edges which may be part of a solution and the edges which can definitely not be part of a solution we have the following definition:

▶ **Definition 11.** *An edge $e \in E$ is **deletable** if $G - e$ is κ-connected. If an edge is not deletable then it is **undeletable**.*

Clearly every edge of a solution $F \subseteq E$ is deletable in G. Let the set of deletable edges of G be denoted $\mathsf{del}(\mathbf{G})$, and denote the remaining edges by $\mathsf{undel}(G)$. It means that for a solution $F \subseteq E$ we have $F \subseteq \mathsf{del}(G)$. Instead of working directly with deletable and undeletable edge we will be working with a subset with the property that $\mathcal{R} \subseteq \mathsf{del}(G)$, such that if there exists a solution F then there also exists a solution $F' \subseteq \mathcal{R}$. The idea behind the algorithm is that we will either find a solution or shrink the set $\mathcal{R}$ until $\mathcal{R}$ is small enough that we can apply a brute force algorithm to determine if a solution exists or no solution exists.

▶ **Definition 12.** *A set $\mathcal{R} \subseteq \mathsf{del}(G)$ is a set of **relevant** edges if one of the following is true.*
- *There exists a solution $F \subseteq \mathcal{R}$.*
- *There exists no solution to the problem.*

Now we have the following relation between a deletable edge $e \in E$ and a nearby vertex-separators with respect to e.

▶ **Proposition 13.** *Given a graph $G = (V, E)$ and a deletable edge $e \in E$ it holds that every nearby vertex-separator with respect to e in G is of size at least $\kappa + 1$.*

3 Overview of the Algorithm

In this section we present an overview of our algorithm, that highlights the main ideas and techniques of this paper. Broadly we follow the approach of Bang-Jensen et al. [1], who studied the edge-connectivity version of the problem in both graphs and digraphs, and also the vertex-connectivity version in digraphs. However we develop some non-trivial new ideas and methods for vertex connectivity in undirected graphs. Due to limited space, the proofs and some technical details have been omitted from this extended abstract; they will be presented in the full version of the paper.

The starting point of our algorithm (similar to Bang-Jensen et al [1]) is the notion of a *deletable edge*. Let $\mathsf{del}(G)$ be the set of all deletable edges in G. Note that it is computable in polynomial time. It is clear that any solution F to this instance is a subset of $\mathsf{del}(G)$. Here, by a *solution to* (G, ℓ) we mean a subset of ℓ edges, F, such that $G - F$ is κ-connected. Note that, if $|\mathsf{del}(G)|$ itself is upper-bounded, by say $49\kappa^2\ell^4$, then a simple brute-force algorithm can find a solution in the time $2^{\mathcal{O}(\kappa^2\ell^4)} \cdot n^{\mathcal{O}(1)}$. Therefore, we may assume that $|\mathsf{del}(G)| \geq 49\kappa^2\ell^4$.

Consider the effect of removing a deletable edge $e \in \mathsf{del}(G)$ from G. Observe that a number of edges in $\mathsf{del}(G)$ could become *undeletable* in $G - e$, i.e. removing any of these edges in $G - e$ will violate the κ-connectivity constraint. Let $\mathcal{I}_e^G = \mathsf{del}(G) \setminus \mathsf{del}(G - e)$ denote the set of all deletable edges in G that become undeletable in $G - e$. We drop the superscript when the graph is clear from context. Now consider the following simple greedy algorithm based on the notion of deletable edges:

> If there is a deletable edge e in the current instance (G, ℓ), find the one minimizing $|\mathcal{I}_e|$. Then recursively solve the instance $(G - e, \ell - 1)$ to obtain a solution S' to it. Finally output the solution $S = S' \cup \{e\}$. Otherwise, output that there is no solution.

Observe that this algorithm succeeds only if in each sub-instance, the set $\mathcal{I}_e$ is not too large (e.g. $\mathcal{O}(\kappa^2\ell^3)$). In other words, only a bounded number of deletable edges should become undeletable in each recursive call. Then assuming that $\mathsf{del}(G)$ was sufficiently large (e.g. at least $\mathcal{O}(\kappa^2\ell^4)$) at the beginning, this greedy algorithm produces a solution to (G, ℓ) in polynomial time.

The remaining case is when there is a sub-instance G' where, for any deletable edge $e' \in \mathsf{del}(G')$ $|\mathcal{I}_{e'}^{G'}| > 100\kappa^2\ell^3$, that is a large number of deletable edges become undeletable on removing e' from G'. We pick such an edge $e' \in \mathsf{del}(G')$ and consider the pair (G', e') and

analyze their structural properties. Our main technical result, simplified, is an algorithm $\mathcal{A}_{Irr}$ that given G, G', e', where G' is a κ-connected subgraph G' of G and $e' \in \mathsf{del}(G')$ such that $|\mathcal{I}_{e'}^{G'}| > 100\kappa^2\ell^2$, either finds a solution to (G, ℓ), or identifies a new irrelevant edge $e'' \in \mathcal{I}_{e'}^{G'}$ for the instance (G, ℓ). Here, an *irrelevant edge* refers to a deletable edge $e'' \in \mathsf{del}(G)$ such that there is a solution $F \subseteq \mathsf{del}(G) \setminus e''$. If we mark e'' as irrelevant, then it is treated just like an undeletable edge, and it remains in the graph.

The algorithm runs over many iterations, where in each iteration we either find a solution or find a new irrelevant edge. We maintain a set of *relevant* edges $\mathcal{R}$ which is initially $\mathsf{del}(G)$, and update it over a sequence of iterations until either a solution is found, or we have rule out the existence of a solution. Note that we ensure that $\mathcal{R}$ is always a subset of the deletable edges. Further, the definition of the set $\mathcal{I}_e$ now becomes $\mathcal{R} \setminus \mathsf{del}(G - e)$, since we are only interested in solutions formed with relevant edges. Therefore our updated algorithm for p-κ-VCSS is as follows:

> Let (G, ℓ) be the input instance of p-κ-VCSS, and let $\mathcal{R} \subseteq \mathsf{del}(G)$ be the set of relevant edges. We first apply the above greedy algorithm to (G, ℓ), and output a solution if one is found.
> Otherwise the above greedy algorithm fails to find a solution to (G, ℓ), therefore we find a sub-instance G' and pick an arbitrary edge $e' \in \mathsf{del}(G')$ such that $\mathcal{I}_{e'}$ is large. Then, apply the algorithm $\mathcal{A}_{Irr}$ to $(G, G', e', \mathcal{R})$. It either finds a solution F to (G, ℓ), or it finds a new irrelevant edge $e'' \in \mathcal{I}_{e'} \subseteq \mathcal{R}$ for (G, ℓ). In the first case, we output the solution found by $\mathcal{A}_{Irr}$. In the second case, start a new iteration on the instance (G, ℓ) with $\mathcal{R} - e''$ as the new set of relevant edges.

Observe that we have at most $|E|$ iterations of the above algorithm. Hence, the running time of the above algorithm is $(t_{\mathcal{A}_{Irr}} + 2) \cdot n^{\mathcal{O}(1)}$, where $t_{\mathcal{A}_{Irr}}$ denotes the running time of the algorithm $\mathcal{A}_{Irr}$. We prove that $t_{\mathcal{A}_{Irr}} = 2^{\mathcal{O}(\kappa^2\ell^4)} \cdot n^{\mathcal{O}(1)}$ where $n = |V|$.

Let us also address another special case of the problem that can be solved in polynomial time. It leads to a bound on the number of relevant edges incident on any single vertex. This is required for our implementation of $\mathcal{A}_{Irr}$. We begin by observing the following.

▶ **Observation 14.** *Given a κ-connected graph $G = (V, E)$, a set of relevant edges $\mathcal{R} \subseteq E$, and an edge $e = uv \in \mathcal{R}$. In $G - e$ every set of κ internally disjoint paths from u to v, $\mathcal{P} = \{P_1, P_2, \ldots, P_\kappa\}$, will use all of the irrelevant edges with respect to e, that is, $\mathcal{I}_e^G \subseteq \cup_{i \in \{1,\ldots,\kappa\}} E(P_i)$.*

▶ **Lemma 15.** *Given a κ-connected graph $G = (V, E)$ and a set of relevant edges $\mathcal{R}$. If there exists a vertex $u \in V$ such that $|\delta(u) \cap \mathcal{R}| \geq (\kappa + 1) \cdot l$, then a solution exists and it can be found in $O((|E| + |V|)^{O(1)})$ time.*

In the following subsections we give an overview of main ideas of the algorithm $\mathcal{A}_{Irr}$. For simplicity, we assume that $G = G'$, and we have an edge $e \in \mathcal{R} \subseteq \mathsf{del}(G)$ such that $|\mathcal{I}_e| > 100\kappa^2\ell^3$. In the general case, where G' is a proper subgraph of G, our methods are identical with one extra step for handling the edges in $E(G) \setminus E(G')$. We construct a Wheel decomposition of the graph G' and then we lift it back to G (see Lemma 22).

The Wheel Decomposition

The key conceptual idea behind the algorithm $\mathcal{A}_{Irr}$, is what we call the *Wheel decomposition* of a graph; the name is derived from the decomposition structure. See Figure 2.

▶ **Definition 16.** *Let $r \geq 3$ and α be integers. We say that G admits an α-wheel decomposition $(\mathcal{W}, r)$ if there is an edge partition $\mathcal{W} = \{X_0, X_1, \ldots, X_{r-1}\}$ of $E(G)$ into nonempty subsets, where the subsets are indexed cyclically, such that for every $i \in \{0, \ldots, r-1\}$ the following holds.*

1. $|\mathcal{S}(X_i)| = \alpha$
2. $\mathcal{S}(X_i) = \mathcal{D}(X_{i-1}, X_i) \cup \mathcal{D}(X_i, X_{i+1})$,
3. $|\mathcal{D}(X_{i-1}, X_i)| = |\mathcal{D}(X_i, X_{i+1})|$,
4. $\mathcal{D}(X_{i-1}, X_i) \cap \mathcal{D}(X_i, X_{i+1}) = \mathcal{S}(X_i) \cap \mathcal{S}(X_j)$ *for every* $j < i-1$ *and* $j > i+1$.

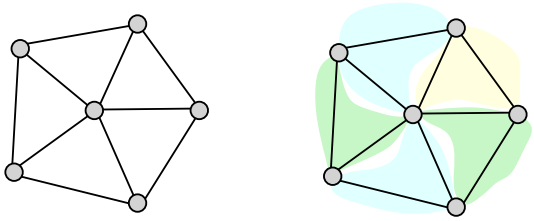

Figure 2 A simple example of a Wheel decomposition of a graph; each part contains two edges and the center vertex is the only middle vertex of this wheel decomposition.

Observe that Property 4, ensures that for every vertex $v \in V(G)$ either $\delta(v)$ contains an edge from every part of $\mathcal{W}$, or from at most two consecutive parts of $\mathcal{W}$, where X_{r-1} and X_0 are taken to be consecutive giving $\mathcal{W}$ a cyclic order. The vertices of the first kind, i.e. those that have an edge in every part of $\mathcal{W}$ are called the *middle vertices* of $\mathcal{W}$. We think of these vertices as the hub of a wheel whose ring is formed by the non-middle vertices of $\mathcal{W}$. Note the highly symmetrical properties of the vertex-cuts induced by the parts of $\mathcal{W}$. These are extensively applied in our proofs.

For the algorithm $\mathcal{A}_{Irr}$, we shall associate a κ-Wheel decomposition $\mathcal{W}$ with a subset of edges $E^+ \subseteq \mathcal{R}$. Recall that the input to this algorithm consists of a tuple $(G, \mathcal{R}, G', e')$, and in the simplified case that we are currently considering $G = G'$. Further, we have that $\mathcal{I}_{e'} = \mathcal{R} \setminus \mathsf{del}(G - e')$ contains at least $100\kappa^2\ell^3$ edges. We will compute a subset E^+ of $\mathcal{I}_e$, and use a Wheel decomposition to identify an irrelevant edge in E^+. Towards this we first recall the notion of a nearby vertex separator. We then have the following definition.

▶ **Definition 17.** *Let α be an integer. Let G be a graph and let $E^+ = \{e_0, e_1, \cdots, e_{|U|-1}\} \subseteq E(G)$ be a subset of edges. We say that G admits an* ***edge-restricted α-wheel decomposition*** *$(\mathcal{W}, E^+)$ if there is an α-wheel decomposition $(\mathcal{W}, |E^+|)$ such that for every $i \in \{0, \ldots, |E^+| - 1\}$;*

- $e_i \in X_i$,
- $\mathcal{S}(X_i)$ *and* $\mathcal{S}(X_{i+1})$ *are both nearby vertex-separators with respect to* e_i.

▶ **Observation 18.** *Let G be a graph and let $E^+ = \{e_0, e_1, \ldots, e_{r-1}\} \subseteq \mathcal{R}$ be a subset of relevant edges. Let $(\mathcal{W}, E^+)$ be an edge-restricted α-Wheel decomposition of G, for some integer α, where $\mathcal{W} = \{X_0, X_1, \ldots, X_{r-1}\}$. Then for each edge $e_i = u_i v_i \in E^+$ such that $e_i \in X_i$, we have that $\delta(u_i) \subseteq X_i$ and $\delta(v_i) \setminus \{e_i\} \in X_{i+1}$. Here X_r denotes the set X_0.*

The key idea is the fact that both $\mathcal{S}(X_i)$ and $\mathcal{S}(X_{i+1})$ are nearby vertex separators for e_i. This means that both $\mathcal{S}(X_{i+1})_{G-e_i}$ and $\mathcal{S}(X_i)_{G-e_i}$ are $u_i - v_i$ vertex separators. Hence $\delta(u_i) \setminus e_i \subseteq X_i \setminus e_i$ and $\delta(v_i) \setminus e_i \subseteq \overline{X_i}$. We can draw similar conclusions about X_{i+1}, and then arrive at these observations. It is immediate from Observation 18 that both endpoints of an edge $e_i \in E^+$ must be non-middle vertices of the Wheel decomposition $(\mathcal{W}, E^+)$. This

fact will be important in our proofs. However, we must also deal with another issue. Suppose that we have a $(\kappa+1)$-Wheel decomposition $(\mathcal{W}, E^+)$ and we wish to mark an edge $e_i \in E^+$ as irrelevant. To argue the correctness, we have to show that there is a solution that excludes e_i. Towards this, given some solution F to (G, ℓ) that contains e_i, we construct another solution F' that excludes e_i. The construction of F' in our proofs must not only exclude e_i but every edge in $E'_i = F \cap E(\mathcal{D}(X_i, X_{i+1})) \setminus Z$, where Z denotes the set of middle vertices of the Wheel decomposition $(\mathcal{W}, E^+)$; this is a constraint of our arguments. These edges will be replaced by another set of edges contained in $E(\mathcal{D}(X_j, X_{j+1})) \setminus Z$ for some $j \neq i$. To describe the construction of F' formally we require the following definition.

▶ **Definition 19.** *Given a κ-connected graph G, and a set of relevant edges $\mathcal{R} \subseteq E(G)$. If G admits an edge-restricted $(\kappa+1)$-wheel decomposition $(\mathcal{W}, E^+)$ where $E^+ \subseteq \mathcal{R}$ then a* ***friendly set of edges for $e_i \in E^+$*** *is a set of edges $E_i \subseteq \mathcal{R}$ such that*

1. *$e_i \in E_i$,*
2. *every edge $e \in E_i$ has at least one of it's endpoints in $\mathcal{D}(X_i, X_{i+1})$ and none of the endpoints of e are middle vertices,*
3. *$G - E_i$ is κ-connected,*

and E_i has the maximum cardinality among all edge subsets fulfilling these three criteria.

Observe that a friendly set of edges E_i for an edge $e_i \in E^+$ gives an upper-bound on the set E'_i described above. It suggests that choosing e_i that minimizes $|E_i|$ is the most likely candidate for a new irrelevant edge. The following lemma gives us a way of a computing friendly set of edges for a given wheel decomposition $(\mathcal{W}, E^+)$.

▶ **Lemma 20.** *Given a κ-connected graph G, a set of relevant edges $\mathcal{R}$, and an edge-restricted $(\kappa+1)$-wheel decomposition $(\mathcal{W}, E^+)$ of G where $E^+ \subseteq \mathcal{R}$, it is possible to find a family of friendly sets $\mathcal{E} = \{E_i | i \in \{0, \ldots, |E^+|\}\}$ in $O(2^{O(d\cdot(\kappa+1))} \cdot |G|^{O(1)})$ time, where d is the maximum number of relevant edges incident on a vertex $v \in V(G)$.*

Note that this lemma assumes that the number of relevant edges incident on any vertex is bounded by some number d. Here we require Lemma 15, where we have shown that if there were a vertex v that had $(\kappa+1)\cdot\ell$ relevant edges incident on it, then we can compute a solution in polynomial time. Hence, we can assume that $d < (\kappa+1)\cdot\ell$, and therefore compute a friendly set of edges in $2^{\mathcal{O}((\kappa+1)^2\cdot l)} \cdot n^{\mathcal{O}(1)}$ time. This is suitable for an FPT algorithm.

With all these definitions and lemmas in hand, the process of identifying a new irrelevant edge in an instance $(G, \mathcal{R}, \ell)$ will be as follows. Using the steps described in the following sub-section, we will arrive at a subset of relevant edges $E^+ \subseteq \mathcal{R}$ containing more than $6\ell+\kappa+2$ edges and an edge-restricted $(\kappa+1)$-Wheel decomposition $(\mathcal{W}, E^+)$ in polynomial time. Next, we will compute a friendly set of edges for every $e_i \in E^+$. Here we apply Lemma 20, along with the bound on the number of relevant edges incident to a vertex due to Lemma 15. This yields a friendly set of edges, E_i for each edge in $e_i \in E^+$, in total time $2^{\mathcal{O}((\kappa+1)^2\cdot l)} \cdot n^{\mathcal{O}(1)}$. Finally, we mark the edge in $e_i \in E^+$ with the smallest friendly set of edges as irrelevant. The correctness of the last step is from the following lemma, which is one of our main technical results.

▶ **Lemma 21.** *Given a graph $G = (V, E)$, a set of relevant edges $\mathcal{R}$, an edge-restricted $(\kappa+1)$-wheel decomposition $(\mathcal{W}, E^+)$ of G with $E^+ \subseteq \mathcal{R}$ where $|E^+| \geq 6l+\kappa+2$, and a family of friendly sets $\mathcal{E}$ for the edges in E^+ such that $E_s \in \mathcal{E}$ has the minimum cardinality of all the sets in $\mathcal{E}$, it holds that $\mathcal{R} - e_s$ is a set of relevant edges in G.*

To prove this lemma we consider a hypothetical solution F such that $e_s \in F$ and then we construct another solution F' that excludes e_s. This then implies that e_s is irrelevant. Let $\mathcal{W} = \{X_0, X_1, \ldots, X_{6\ell+\kappa+1}\}$. Since $|F| = \ell$, there must exist an index $i \in \{1, \ldots, 6\ell+\kappa+2\}$

such that for every $j \in \{-2, \ldots, 3\}$, $X_{i+j} \cap F = \emptyset$. Let $E'_s = F \cap \mathcal{D}(X_i, X_{i+1})$ and note that $e_s \in E'_s$. Let $F' = (F \setminus E'_s) \cup E_i$. We claim that F' is also a solution. It is clear that $|F'| \geq |F|$ by our choice of e_s. To argue that $G - F'$ is κ-vertex connected, we give an argument based on (hypergraph) cut submodularity. While the actual argument is quite long and non-trivial, the essence of it is that if there were a $\kappa - 1$ cut in $G - F'$, then using submodularity we can trim it down to a $\kappa - 1$ cut that must exist in $G - F$. This is a contradiction since F is a solution, i.e. $G - F$ is κ-connected. Here our arguments make extensive use of the highly symmetric properties of the cuts $\mathcal{S}(X_i)$.

The next lemma allows us to lift a Wheel decomposition from a subgraph G' back to a graph G. This is required for the case where our simplifying assumption that $G' = G$ doesn't hold. In that case, $G = G' + F'$ and we have a Wheel decomposition in G' that must be converted into a decomposition in G.

▶ **Lemma 22.** *Given a graph G', a nonempty set of edges $F' \subseteq V(G') \times V(G')$, an edge-restricted $(\kappa + 1)$-wheel decomposition $(\mathcal{W}' = \{X'_0, X'_1, \cdots, X'_{|\mathcal{W}'|-1}\}, E' = \{e'_0, e'_1, \cdots, e'_{|\mathcal{W}|'-1}\})$ of G' such that $|\mathcal{W}'| \geq 2 \cdot |F'| \cdot (r + 1) + 1$ for an integer $r \geq 4$, it holds that in polynomial time it is possible to find an edge-restricted $(\kappa + 1)$-wheel decomposition $(\mathcal{W}', E^+)$ of $G = G' + F'$ such that $|\mathcal{W}'| = r$ and $E^+ \subseteq E'$.*

The proof of this lemma essentially attempts to "merge" those parts $\{X'_i\}$ that are damaged by adding F'. The details are presented in the full-version.

Wheel Decomposition for p-κ-VCSS

In this section we give an overview of the next part of the algorithm $\mathcal{A}_{Irr}$ that computes a Wheel decomposition with certain useful properties. The input is a κ-connected graph G on n vertices, an integer ℓ and a set of relevant edges $\mathcal{R}$ and an edge $e \in \mathcal{R}$ such that $\mathcal{I}_e = \mathcal{R} \setminus \mathsf{del}(G - e)$ contains at least $\kappa\ell \cdot 2(h + 3)$ edges. Here $h \geq 3$ denotes the number of parts in our Wheel decomposition, and it is an integer whose value is roughly $O(\kappa\ell^2)$ in our final algorithm. We compute a edge-restricted Wheel decomposition $(\mathcal{W}, E^+)$ where $E^+ \subseteq \mathcal{I}_e$ using the following lemma, which is our second main technical result.

▶ **Lemma 23.** *Given a κ-connected graph $G = (V, E)$, an integer $h \geq 3$, an edge $e = uv \in \mathcal{R}$ where $\mathcal{R}$ is a relevant set of edges for G, such that $|\mathcal{I}_e| \geq \kappa \cdot \ell \cdot 2(h + 3)$, in polynomial time G it is possible to either:*

1. *find a set of edges $F \subseteq E$ such that $G - F$ is κ-connected and $|F| = \ell$, or*
2. *find a set $E^+ \subseteq \mathcal{I}_e$ such that $|E^+| = h$ and an edge-restricted $(\kappa+1)$-wheel decomposition $(\mathcal{W}, E^+)$ of G.*

Our proof of this lemma relies on several structural properties of e and $\mathcal{I}_e$. In essence, we first identify a suitable subset of edges in $\mathcal{I}_e$ that can lead to the required Wheel decomposition, and then prove this fact. The first ingredient is the following. Let $e = uv$ and since $e \in \mathcal{R}$, the graph $G - e$ is κ-connected. Then we argue that for any system of κ internally disjoint paths from u to v in $G - e$, $\mathcal{P}$, every edge in $\mathcal{I}_e$ occurs in one of the paths in $\mathcal{P}$. This is because if some edge $e' \in \mathcal{I}_e$ was excluded from $\mathcal{P}$, then one can show that $G - e - e'$ must also be κ-connected. From this statement we can conclude that one of the paths in $\mathcal{P}$ must have at least $|\mathcal{I}_e|/\kappa \geq \ell \cdot 2(h + 3)$ edges from $\mathcal{I}_e$, and let $P \in \mathcal{P}$ be such a path. We say that this path P satisfies the *intersection property* with respect to e. Observe that by a simple max-flow computation we can find this path P in polynomial time. Let $\{e_1, e_2, \ldots, e_{\ell \cdot 2(h+3)}\} \subseteq E(P) \cap \mathcal{I}_e$ where the edges are in sequence of traversing P from u to v. Our objective is to show that a subset of these edges can lead us to the required Wheel decomposition.

Let $E' = \{e_i \mid i = 1 \mod 2(h+3)\}$, and let r be the least integer such that $e_r \in E'$ and $G' = G - \{e_i \mid i = 1 \mod 2(h+3) \text{ and } i \leq r\}$ is *not* κ-connected. If $r \geq \ell + 1$, then clearly $\{e_i \mid i = 1 \mod 2(h+3) \text{ and } i < r\}$ is the required solution to $(G, \ell, \mathcal{R})$. Therefore, we assume $r \leq \ell$, and let $E'' = \{e_i \mid i = 1 \mod 2(h+3) \text{ and } i < r\}$. It is clear that $G - E''$ and $G - e_r$ are κ-connected, but $G' = G - E'' - e_r$ is not.

Let us fix a vertex-separator $\mathcal{S}(U')_{G'}$ of size less than κ in G', where $U' \subseteq E(G')$. It is clear that $\mathcal{S}(U')_{G'}$ separates the vertex pairs u_r, v_r and u_j, v_j where $e_r = u_r v_r$ and $e_j = u_j v_j \in E''$ for some $j < r$. Let us fix the largest such j. Note that $r - j \geq 2(h+3)$ and hence $r \geq 2(h+3)$. Let $E^* = \{e_j, e_r\} \cup \{e_{r+2i-2(h+2)} \mid 0 \leq i \leq h+1\}$. Finally, we define the edge subset E^+ which contains h edges, that forms one half of our Wheel decomposition $(\mathcal{W}, E^+)$ as follows:

$$E^+ = E^* \setminus \{e_j, e_{r-2(h+2)}, e_{r-2}, e_r\}$$

We remark that the choice of E^+ may seem rather arbitrary, however it was chosen to encode several useful properties that are required for our proof. These properties are applied in the construction of the edge partition $\mathcal{W}$, which forms the other half of our decomposition.

For convenience, let us rename the edges in E^* as $s_i \leftarrow e_{r+2i-2(h+2)}$ and further let $s = e_j$ and $t = e_r$. The following lemma is our next key ingredient, that computes a family of vertex cuts, defined via edge-subsets, corresponding to E^*. These cuts will be used to define $\mathcal{W}$.

▶ **Lemma 24.** *Given a graph $G = (V, E)$, an edge $e = uv \in \mathcal{R}$ and a path P between u and v fulfilling the intersection property such that $\mathcal{I}_e \cap P \neq \emptyset$, in polynomial time it is possible to find a family of cuts $\mathcal{C} = \{C_1, C_3 \cdots, C_{|\mathcal{I}_e \cap P|}\}$ such that for each $e_i \in \mathcal{I}_e \cap E(P)$,*

1. *$e_i \in C_i$, $e \in \overline{C_i}$, and $\mathcal{S}(C_i)$ is a nearby vertex-separator with respect to both e_i and e.*
2. *v_i is the only internal vertex of P in $\mathcal{S}(C_i)$.*
3. *$|\mathcal{S}(C_i)| = \kappa + 1$,*
4. *For every $e_j = u_j v_j$ where $1 < j < i$, $\delta(u_j), \delta(v_j) \subseteq C_i$*
5. *For every $e_j = u_j v_j$ where $j \geq i + 2$, $\delta(u_j), \delta(v_j) \subseteq \overline{C_i}$*

Further $C_i \subset C_j$ for any $j \geq i + 2$.

Proving this lemma is not too difficult, although it requires working out some details. The idea is to start with a collection of arbitrary κ-cuts in $G - e$ containing an endpoint of these edges, and further exploiting the membership of e_i in $\mathcal{I}_e$. These cuts are "untangled" using the submodularity of hypergraph cuts, to obtain that last containment property. Similar ideas were used by Bang-Jensen et al [1].

Applying the above lemma to G, e and P we obtain a collection of cuts, $\mathcal{C} = \{C_r, C_0, C_1, \ldots C_{h+1}, C_t\}$ corresponding to the edges in $E^* = \{s, s_0, s_1, \ldots, s_{h+1}, t\}$. Note that these cuts satisfy all the properties of the above lemma, and in particular $C_i \subset C_j$ for any $i < j$ in this collection. We can now define the edge-partition of our Wheel decomposition as follows:

$$\mathcal{W} = \{X_1 = E \setminus (C_h \cap \overline{C_1} \cap U'\} \cup \{X_i = C_i \cap \overline{C_{i-1}} \cap U' \mid 2 \leq i \leq h\}$$

Finally, to prove that $\mathcal{W}$ is indeed the required Wheel decomposition, we prove it's properties one by one. The main tool that we use is once again submodularity of cuts. In particular, from the definition of $\mathcal{W}$, some intutition can be derived. The cut induced by U' is of size $\kappa - 1$ in G'. This cut splits each C_i into two pieces, and further because of the containment property of $\mathcal{C}$, the parts X_i are defined as the portion of $C_i \setminus C_{i-1}$ in U'. The part X_1 simply contains the remaining edges. The various symmetric properties of the cuts induced by each X_i are consequences of the choice of E^*, via submodularity of cuts.

Putting it all together: An algorithm for p-κ-VCSS

We collect all the previous results to arrive at the following algorithm for p-κ-VCSS:

1. If the induced graph $G' = (V, \mathcal{R})$ contains a vertex with degree at least $(\kappa+1)\cdot l$ then use the algorithm of Lemma 15 to find a solution and return it.
2. If $|\mathcal{R}| \leq 49\kappa^2 l^4$, then for every subset $F' \subseteq \mathcal{R}$ of size l determine if F' is a solution. If F' is a solution, return F' as a solution. If none of the subsets F' is a solution, return that no solution exists.
3. Choose an edge $f_i \in \mathcal{R}_i$, and let $\mathcal{I}_{f_i} = \mathcal{R}_i \setminus \mathsf{del}(G_i \setminus f_i)$:
   a. If $|\mathcal{I}_{f_i}| \geq \kappa \cdot l \cdot 2(2i\cdot(6l+\kappa+3)+4)$ then by Lemma 23
      i. we can either find a solution F' for G_i of size l, and return F' as a solution,
      ii. or we can find a set $E^+ \subseteq \mathcal{I}_e$ such that $|E^+| = 2i\cdot(6l+\kappa+3)+1$ and an edge-restricted $(\kappa+1)$-wheel decomposition $(\mathcal{W}, E^+)$ for G_i.
         A. If $F \neq \emptyset$ then given G_i, F, and $(\mathcal{W}, E^+)$ by Lemma 22 find an edge-restricted $(\kappa+1)$-wheel decomposition $(\mathcal{W}', E')$ of $G = G_i + F$ such that $E' \subseteq E^+$ and $|E'| \geq 6l+\kappa+2$.
         B. Otherwise, if $F = \emptyset$ then set $(\mathcal{W}', E') = (\mathcal{W}, E^+)$ to be an edge-restricted $(\kappa+1)$-wheel decomposition of $G = G_i$.
         C. Given the set of relevant edges $\mathcal{R}$, and $(\mathcal{W}', E')$ use the algorithm from Lemma 20 to find the friendly sets for every edge $e_j \in E'$.
         D. Find the edge $e_j \in E'$ for which the corresponding friendly set E_j has the smallest cardinality among all the friendly sets. Now set $\mathcal{R} := \mathcal{R} \setminus e_j$ and go to step 2. The correctness of this step follows from Lemma 21.
   b. Otherwise, set $G_{i+1} = G_i - f_i$, $\mathcal{R}_{i+1} = \mathcal{R} \cap \mathsf{del}(G_{i+1})$, and $F := F \cup f_i$. If $|F| \geq l$ return F as a solution.

This algorithm proves Theorem 1. The correctness follows from the correctness of each lemma used in the above. For the running time, observe that Step 1 runs in polynomial time. The next steps are repeated at most $|E(G)|$ times, for each iteration, where we shrink the set of relevant edges $\mathcal{R}$ until Step 2 is finally applicable. Note that Step 2 runs in time $2^{\mathcal{O}(\kappa^2 \ell^4)} \cdot n^{\mathcal{O}(1)}$. One of Steps 3(a)i and 3(a)ii is run in each iteration, which is decided in polynomial time by Lemma 23. In the second case, we obtain a Wheel decomposition $(\mathcal{W}, E^+)$ in Step 3(a)iiC we spend $2^{\mathcal{O}((\kappa+1)^2\cdot l)} \cdot n^{\mathcal{O}(1)}$ time computing friendly sets of edges for E^+. We then mark an irrelevant step, and proceed to the next iteration. A straightforward calculation gives the claimed running time of the algorithm.

Due to space constraints, several proofs were omitted. Complete details may be found in the full version of the paper, in the appendix. Finally, we also thank anonymous reviewers for their suggestions on improving this paper.

References

1 Jørgen Bang-Jensen, Manu Basavaraju, Kristine Vitting Klinkby, Pranabendu Misra, MS Ramanujan, Saket Saurabh, and Meirav Zehavi. Parameterized algorithms for survivable network design with uniform demands. In *Proceedings of the 29th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2838–2850. SIAM, 2018.

2 Jørgen Bang-Jensen and Anders Yeo. The minimum spanning strong subdigraph problem is fixed parameter tractable. *Discrete Applied Mathematics*, 156(15):2924–2929, 2008.

3 Jørgen Bang-Jensen and Gregory Gutin. *Digraphs. Theory, Algorithms and Applications*. Springer, 2002.

4 Manu Basavaraju, Fedor V Fomin, Petr Golovach, Pranabendu Misra, MS Ramanujan, and Saket Saurabh. Parameterized algorithms to preserve connectivity. In *Proceedings of the 41st International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 800–811. Springer, 2014.

5 Jaroslaw Byrka, Fabrizio Grandoni, and Afrouz Jabal Ameli. Breaching the 2-approximation barrier for connectivity augmentation: a reduction to steiner tree. In Konstantin Makarychev, Yury Makarychev, Madhur Tulsiani, Gautam Kamath, and Julia Chuzhoy, editors, *Proccedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020, Chicago, IL, USA, June 22-26, 2020*, pages 815–825. ACM, 2020. doi:10.1145/3357713.3384301.

6 Tanmoy Chakraborty, Julia Chuzhoy, and Sanjeev Khanna. Network design for vertex connectivity. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing (STOC)*, pages 167–176, 2008.

7 Joseph Cheriyan and László A Végh. Approximating minimum-cost k-node connected subgraphs via independence-free graphs. *SIAM Journal on Computing*, 43(4):1342–1362, 2014.

8 Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer Science & Business Media, 2015.

9 Andreas Emil Feldmann, Anish Mukherjee, and Erik Jan van Leeuwen. The parameterized complexity of the survivable network design problem. In *Symposium on Simplicity in Algorithms (SOSA)*, pages 37–56. SIAM, 2022.

10 Fedor V. Fomin, Petr A. Golovach, William Lochet, Pranabendu Misra, Saket Saurabh, and Roohani Sharma. Parameterized complexity of directed spanner problems. In Yixin Cao and Marcin Pilipczuk, editors, *15th International Symposium on Parameterized and Exact Computation, IPEC 2020, December 14-18, 2020, Hong Kong, China (Virtual Conference)*, volume 180 of *LIPIcs*, pages 12:1–12:11. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.IPEC.2020.12.

11 Greg N Frederickson and Joseph Ja'Ja'. Approximation algorithms for several graph augmentation problems. *SIAM Journal on Computing*, 10(2):270–283, 1981.

12 Gregory Gutin, M.S. Ramanujan, Felix Reidl, and Magnus Wahlström. Path-contractions, edge deletions and connectivity preservation. *Journal of Computer and System Sciences*, 101:1–20, 2019. doi:10.1016/j.jcss.2018.10.001.

13 Kamal Jain. A factor 2 approximation algorithm for the generalized steiner network problem. *Combinatorica*, 21(1):39–60, 2001.

14 Samir Khuller. Approximation algorithms for finding highly connected subgraphs. In *Approximation algorithms for NP-hard problems*, pages 236–265. PWS Publishing Co., 1996.

15 Kristine Vitting Klinkby, Pranabendu Misra, and Saket Saurabh. Strong connectivity augmentation is FPT. In Dániel Marx, editor, *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, January 10 – 13, 2021*, pages 219–234. SIAM, 2021. doi:10.1137/1.9781611976465.15.

16 Yusuke Kobayashi. Np-hardness and fixed-parameter tractability of the minimum spanner problem. *Theor. Comput. Sci.*, 746:88–97, 2018. doi:10.1016/j.tcs.2018.06.031.

17 Yusuke Kobayashi. An FPT algorithm for minimum additive spanner problem. In Christophe Paul and Markus Bläser, editors, *37th International Symposium on Theoretical Aspects of Computer Science, STACS 2020, March 10-13, 2020, Montpellier, France*, volume 154 of *LIPIcs*, pages 11:1–11:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.STACS.2020.11.

18 Guy Kortsarz, Robert Krauthgamer, and James R. Lee. Hardness of approximation for vertex-connectivity network design problems. *SIAM Journal on Computing*, 33(3):704–720, 2004. doi:10.1137/S0097539702416736.

19 Guy Kortsarz and Zeev Nutov. Approximating minimum cost connectivity problems. In *Dagstuhl Seminar Proceedings*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2010.

20 Daniel Lokshtanov, M. S. Ramanujan, Saket Saurabh, and Meirav Zehavi. Reducing CMSO model checking to highly connected graphs. In Ioannis Chatzigiannakis, Christos Kaklamanis, Dániel Marx, and Donald Sannella, editors, *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, July 9-13, 2018, Prague, Czech Republic*, volume 107 of *LIPIcs*, pages 135:1–135:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.ICALP.2018.135.

21 Dániel Marx and László A Végh. Fixed-parameter algorithms for minimum-cost edge-connectivity augmentation. *ACM Transactions on Algorithms (TALG)*, 11(4):27, 2015.

22 Zeev Nutov. Approximability status of survivable network problems, 2014.

23 Zeev Nutov. Approximating minimum-cost edge-covers of crossing biset-families. *Combinatorica*, 34(1):95–114, 2014.

24 Vera Traub and Rico Zenklusen. A better-than-2 approximation for weighted tree augmentation. *CoRR*, abs/2104.07114, 2021. arXiv:2104.07114.

25 Vera Traub and Rico Zenklusen. A $(1.5+\epsilon)$-approximation algorithm for weighted connectivity augmentation. *arXiv preprint arXiv:2209.07860*, 2022.

26 László A Végh. Augmenting undirected node-connectivity by one. *SIAM Journal on Discrete Mathematics*, 25(2):695–718, 2011.