Combining Crown Structures for Vulnerability Measures

Katrin Casel
Humboldt Universität zu Berlin, Germany

Tobias Friedrich
Hasso Plattner Institute, University of Potsdam, Germany

Aikaterini Niklanovits
Hasso Plattner Institute, University of Potsdam, Germany

Kirill Simonov
Hasso Plattner Institute, University of Potsdam, Germany

Ziena Zeif
Hasso Plattner Institute, University of Potsdam, Germany

Abstract

Over the past decades, various metrics have emerged in graph theory to grasp the complex nature of network vulnerability. In this paper, we study two specific measures: (weighted) vertex integrity (wVI) and (weighted) component order connectivity (wCOC). These measures not only evaluate the number of vertices that need to be removed to decompose a graph into fragments, but also take into account the size of the largest remaining component. The main focus of our paper is on kernelization algorithms tailored to both measures. We capitalize on the structural attributes inherent in different crown decompositions, strategically combining them to introduce novel kernelization algorithms that advance the current state of the field. In particular, we extend the scope of the balanced crown decomposition provided by Casel et al. [5] and expand the applicability of crown decomposition techniques.

In summary, we improve the vertex kernel of VI from p^3 to $3p^2$, and of wVI from p^3 to $3(p^2 + p^{1.5}p_\ell)$, where $p_\ell < p$ represents the weight of the heaviest component after removing a solution. For wCOC we improve the vertex kernel from $\mathcal{O}(k^2W + kW^2)$ to $3\mu(k + \sqrt{\mu}W)$, where $\mu = \max(k, W)$. We also give a combinatorial algorithm that provides a $2kW$ vertex kernel in fixed-parameter tractable time when parameterized by r, where $r \leq k$ is the size of a maximum $(W + 1)$-packing. We further show that the algorithm computing the $2kW$ vertex kernel for COC can be transformed into a polynomial algorithm for two special cases, namely when $W = 1$, which corresponds to the well-known vertex cover problem, and for claw-free graphs. In particular, we show a new way to obtain a $2k$ vertex kernel (or to obtain a 2-approximation) for the vertex cover problem by only using crown structures.

2012 ACM Subject Classification Theory of computation → Graph algorithms analysis

Keywords and phrases Crown Decomposition, Kernelization, Vertex Integrity, Component Order Connectivity

Digital Object Identifier 10.4230/LIPIcs.IPEC.2024.1

Related Version *Full Version*: `https://doi.org/10.48550/arXiv.2405.02378` [6]

1 Introduction

In the study of graph theory different scales have emerged over the past decades to capture the complex nature of network vulnerability. While the main focus is on the connectivity of vertices and edges, subtle aspects of vulnerability such as the number of resulting components, the size distribution of the remaining components, and the disparity between them are becoming increasingly interesting [3, 13, 14, 16, 15, 20]. Our focus in this study is on two specific ways to measure vulnerability: (weighted) vertex integrity and (weighted) component order connectivity. These measures not only evaluate the number of vertices that need to be removed to decompose a graph into fragments, but also take into account the size of the largest remaining component. Incorporating these aspects provides a more comprehensive understanding of network resilience.

Informally, *vertex integrity* (VI) is a model for the right balance between removing few vertices and keeping small connected parts of a graph. More formally, given a graph $G = (V, E)$ and a number $p \in \mathbb{N}$, the task for VI is to find a set of vertices $S \subseteq V$ such that $|S|$ plus the size of the largest component when removing S from G is at most p. In the vertex weighted version (wVI), the goal is bounding the total weight of the removed vertices plus the weight of the heaviest component by p. This problem was introduced by Barefoot et al. [2] as a way to measure vulnerability of communication networks. Recently, it has drawn the interest of the parameterized complexity community due to its status as a natural parameter that renders numerous NP-hard problems amenable to fixed parameter tractability (FPT). This means that for these problems, solutions can be computed within a time frame represented by $f(p) \cdot n^{\mathcal{O}(1)}$, where f is a computable function [20]. It is interesting to see how vertex integrity relates to other well-known measures of network structure. It imposes greater constraints compared to metrics such as treedepth, treewidth, or pathwidth, as the vertex integrity of a graph serves as an upper bound for these parameters. However, it encompasses a wider range of scenarios compared to vertex cover, where a vertex cover of a graph is an upper bound for its vertex integrity. This makes it a key in understanding how to efficiently solve problems in the world of network analysis.

The measure *component order connectivity* (COC) can be seen as the refined version of VI. Given a graph $G = (V, E)$ and two parameters $k, W \in \mathbb{N}$, the goal of COC is to remove k vertices such that each connected component in the resulting graph has at most W vertices – also known in the literature as the *W-separator problem* or *α-balanced separator problem*, where $\alpha \in (0, 1)$ and $W = \alpha|V|$. In the vertex-weighted version (wCOC), the goal is to remove vertices of total weight at most k such that the weight of the heaviest remaining component is at most W. An equivalent view of this problem is to search for the minimum number of vertices required to cover or hit every connected subgraph of size $W + 1$. In particular, $W = 1$ corresponds to covering all edges, showing that the COC is a natural generalization of the vertex cover problem.

The focus of the paper is on kernelization algorithms tailored for both weighted and unweighted versions of VI and COC when parameterized by p and $k + W$, respectively. Kernelization algorithms can be thought of as formalized preprocessing techniques aimed to reduce optimization problems. Of particular interest in this work are crown decompositions, which are generally used as established structures for safe instance reduction – where "safe" in this case means that any optimal solution to the reduced instance can efficiently be transformed into an optimal solution of the original. Essentially, a crown decomposition partitions the vertex set into distinct components: crown, head, and body. Here, the head acts as a separator between the crown and the body. This structural arrangement becomes useful when specific relationships between the head and the crown are required. Such relationships ultimately enable us to shrink instances by eliminating these designated parts from the graph. The properties of this structural layout, coupled with its existence depending on the instance size, enable the development of efficient kernelization algorithms for different problem domains. Notably, crown decompositions have also recently found utility in approximation algorithms for graph packing and partitioning problems [5]. For further exploration of crown decompositions, including their variations and applications, we recommend the comprehensive survey paper by Jacob et al. [17].

Our methods take advantage of the structural characteristics found in various crown decompositions, leading to the development of new kernelization algorithms that improve the state of the art. In essence, this work expands upon the applications of the balanced crown decomposition introduced by Casel et al. [5], specifically by integrating different crown decompositions into this framework.

Related Work

Recently, the vertex integrity problem (VI) was extensively studied as a structural graph parameter [3, 13, 15, 20]. Gima et al. [14] conducted a systematic investigation into structural parameterizations of computing the VI and wVI which was further extended by Hanaka et al. [16]. Additionally, there are notable results concerning special graph classes [1, 8, 10, 18, 23]. In our context, regarding related work on VI and wVI, Fellows and Stueckle presented an algorithm that solves the problem in $\mathcal{O}(p^{3p}n)$ [11]. This was subsequently improved by Drange et al. [10], even for the weighted case, to $\mathcal{O}(p^{p+1}n)$. In the same paper, they presented the first vertex-kernel of size p^3 for both VI and wVI.

Considering, COC and wCOC, it is unlikely that kernelization algorithms for these problems can be achieved by considering k or W alone in polynomial time. Indeed, $W = 1$ corresponds to the NP-hard vertex cover problem, which shows that W (alone) is not a suitable parameter. For the parameter k, the problem is $W[1]$-hard even when restricted to split graphs [10]. These lower bounds lead to the study of parameterization by $k + W$. The best known algorithm with respect to these parameters finds a solution in time $n^{\mathcal{O}(1)} \cdot 2^{\mathcal{O}(\log(W)\cdot k)}$ [10]. Unless the exponential time hypothesis fails, the authors prove that this running time is tight in the sense that there is no algorithm that solves the problem in time $n^{\mathcal{O}(1)} \cdot 2^{o(\log(W)\cdot \mathrm{OPT})}$. The best known approximation algorithm has a multiplicative gap guarantee of $\mathcal{O}(\log(W))$ to the optimal solution with a running time of $n^{\mathcal{O}(1)} \cdot 2^{\mathcal{O}(W)}$ [21]. In [21], they also showed that the superpolynomial dependence on W may be needed to achieve a polylogarithmic approximation. Using this algorithm as a subroutine, the vertex integrity can be approximated within a factor of $\mathcal{O}(\log(\mathrm{OPT}))$, where OPT is the vertex integrity.

Regarding kernelization algorithms, there is a sequence of results which successively improve the vertex-kernel of COC. The first results came from Chen et al. [7] and Drange et al. [10], who provided kernels of size in $\mathcal{O}(kW^3)$ and $\mathcal{O}(k^2W + kW^2)$, respectively. The result of Drange et al. also holds for wCOC and is the only result for this case. This was improved simultaneously by Xiao [24] as well as by Kumar and Lokshtanov [19] to a $\mathcal{O}(kW^2)$ kernel. These works also provide the first $\mathcal{O}(kW)$ kernels, but with different constants and running times. Kumar and Lokshtanov [19] present a $2kW$ kernel in a running time of $n^{\mathcal{O}(W)}$ by using linear programming (LP) methods with an exponential number of constraints. The runtime can be improved to $2^{\mathcal{O}(W)} \cdot n^{\mathcal{O}(1)}$ as already mentioned in the book of Fomin et al. [12] (Section 6.4.2). Roughly speaking, the idea is to use the ellipsoid method with separation oracles to solve the linear program, where the separation oracle uses a method called color coding to find violated constraints that makes it polynomial in W. Note that if W is a constant this $2kW$ kernel is a polynomial time kernel improving on some previous results. This includes for instance the improvement of the $5k$ kernel provided by Xiao and Kou [25] to a $4k$ kernel for the well-studied P_2-covering problem, where a P_2 is a path with 2 edges. The first linear kernel in both parameters in polynomial time, i.e. an $\mathcal{O}(kW)$ vertex kernel, is presented by Xiao [24], who provides a $9kW$ vertex kernel. Finally, this was improved by Casel et al. [5] to a $3kW$ vertex kernel, which also holds for a more general setting. Namely, to find k vertices in a vertex weighted graph such that after their removal each component weighs at most W. Note that the weights of the separator, i.e. the chosen k vertices, play no role in this problem compared to wCOC. With the exception of the $\mathcal{O}(k^2W + kW^2)$ vertex kernel of Drange et al. [10], all achieved vertex kernels essentially use crown structures.

Our Contribution

The provided running times are based on an input graph $G = (V, E)$. We improve the vertex kernel for VI from p^3 to $3p^2$ in time $\mathcal{O}\left(\log(p)|V|^4|E|\right)$. For wVI, we improve it to $3(p^2 + p^{1.5}p_\ell)$ in time $\mathcal{O}\left(\log(p)|V|^4|E|\right)$, where $p_\ell \leq p$ represents the weight of the heaviest component after removing a solution.

To better explain the results of COC and wCOC, consider the problem of a maximum λ-packing for $\lambda \in \mathbb{N}$. Given a (vertex weighted) graph G, this problem aims to maximize the number of disjoint connected subgraphs, each of size (or weight) at least λ. It is worth noting that the size of a maximum $(W + 1)$-packing serves as a lower bound on the size of an optimal solution of COC (or wCOC), since each element in the packing must contain at least one vertex of it – in terms of linear programming, it is the dual of COC.

For wCOC we improve the vertex kernel of $\mathcal{O}(k^2W + kW^2)$ to $3\mu(k + \sqrt{\mu}W)$ in time $\mathcal{O}\left(r^2k|V||E|\right)$, where $\mu = \max(k, W)$ and $r \leq |V|$ is the size of a maximum $(W + 1)$-packing. For the unweighted version, we provide a $2kW$ vertex kernel in an FPT-runtime of $\mathcal{O}(r^3|V||E| \cdot r^{\min(3r,k)})$, where $r \leq k$ is the size of a maximum $(W + 1)$-packing. Comparing this result with the state of the art, disregarding the FPT-runtime aspect, we improve upon the best-known polynomial algorithm, achieving a kernel of size $3kW$ [5]. A $2kW$ vertex kernel is also presented in [19], albeit with an exponential runtime using linear programming methods, which, as mentioned, can be enhanced to an FPT-runtime regarding parameter W. In contrast, our result is entirely combinatorial and has an FPT-runtime in the parameter of a maximum $(W + 1)$-packing $r \leq k$. It should be noted that, strictly speaking, the $2kW$ vertex kernel of [19] and our work cannot be considered a kernel, given that the runtime dependency is exponential with respect to the parameters W and k, respectively. However, for the sake of simplicity, we will refer to it as a kernel, with an explicit mention of the runtime dependency. As previously stated, note that it is unlikely that an FPT-runtime will be able to solve COC (or wCOC) when considering either W or k alone. We further show that the algorithm computing the $2kW$ vertex kernel for COC can be transformed into a polynomial algorithm for two special cases. The first case arises when $W = 1$, i.e., for the vertex cover problem. Here, we provide a new method for obtaining a vertex kernel of $2k$ (or obtaining a 2-approximation) using only crown decompositions. The second special case is for the restriction of COC to claw-free graphs. Unfortunately, we currently do not know whether COC is hard on claw-free graphs and defer this question to future work.

Regarding these special cases, until 2017, a $2k$ vertex kernel for $W = 1$ was known through crown decompositions, albeit computed using both crown decompositions and linear programming. Previously, only a $3k$ vertex kernel was known using crown decompositions alone. In 2018, Li and Zhu [22] provided a $2k$ vertex kernel solely based on crown structures. They refined the classical crown decomposition, which possessed an additional property allowing the remaining vertices of the graph to be decomposed into a matching and odd cycles after exhaustively applying the corresponding reduction rule. In contrast, our algorithm identifies the reducible structures, which must exist if the size of the input graph exceeds $2k$. Unfortunately, we were unable to transform the FPT-runtime algorithm into a polynomial-time algorithm in general. Nevertheless, we believe that our insights into the structural properties of certain crown decompositions potentially pave the way to achieving this goal.

Lastly, all missing details, i.e. omitted proofs and algorithms as well as more detailed explanations can be found in the extended arxiv version [6].

2 Preliminaries

In this section, we briefly discuss terminology related to graphs and parameterized complexity that we use in the extended abstract (the complete terminology can be found in the extended arxiv version [6]). Additionally, we introduce the crown structures that are utilized throughout this paper.

Graph and Parameterized Terminology

We use a standard terminology for graphs $G = (V, E)$. The following is not necessarily common: For sets of vertex sets $\mathcal{V} \subseteq 2^V$ we abuse notation and use $V(\mathcal{V}) = \bigcup_{Q \in \mathcal{V}} Q$ and $N(\mathcal{V}) := \left(\bigcup_{Q \in \mathcal{V}} N(U)\right) \setminus V(\mathcal{V})$. We define $\mathbb{CC}(G) \subseteq 2^V$ as the connected components of G as vertex sets. Let $G = (V, E, w\colon V \to \mathbb{N})$ be a vertex weighted graph. For $V' \subseteq V$ and $G' \subseteq G$ we define $w(V')$ and $w(G')$ as $\sum_{v \in V'} w(v)$ and $\sum_{v \in V(G')} w(v)$, respectively.

For improved readability, we extend the notation of the inverse function f^{-1}. Henceforth, sometimes it returns a vertex set that is the union of the corresponding vertex sets, which is always clear from the context. For instance, for a function $f\colon V \to V$ on the vertices of a graph $G = (V, E)$, $h \in V$ and $V' \subseteq V$ we use $f^{-1}(h)$ also for $V(f^{-1}(h))$, and $f^{-1}(V')$ for $V(f^{-1}(V'))$, respectively.

For parameterized complexity we use the standard terminology, which is also used, for example, in [9, 12]. Of particular interest in this work are kernelizations, which can be roughly described as formalized preprocessings. More formally, given an instance (I, k) parameterized by k, a polynomial algorithm is called a kernelization if it maps any (I, k) to an instance (I', k') such that (I', k') is a yes-instance if and only if (I, k) is a yes-instance, $|I'| \leq g(k)$, and $k' \leq g'(k)$ for computable functions g, g'.

Crown Decompositions

We present a more general variant of crown decomposition that also captures commonly used crown decompositions or expansions (strictly speaking the relevant parts of it), which we explain in a moment. This novel decomposition can be easily derived from existing results of [5], but to the best of our knowledge has never been used in this form before. (See Figure 1 (left) for an illustration.)

▶ **Definition 1** (Demanded balanced expansion and weighted crown decomposition). *For a graph $G = (A \cup B, E, w)$, a partition $A_1 \cup A_2$ of A, a function $f\colon \mathbb{CC}(G[B]) \to A$, demands $D = \{d_a\}_{a \in A}$ with $d_a \in \mathbb{N}$ for each $a \in A$ and $y \in \mathbb{N}$ the tuple (A_1, A_2, y, f, D) is a demanded balanced expansion* (DBE) if

1. $w(Q) \leq y$ for each $Q \in \mathbb{CC}(G[B])$
2. $f(Q) \in N(Q)$ for each $Q \in \mathbb{CC}(G[B])$
3. $N(f^{-1}(A_1)) \subseteq A_1$
4. $w(a) + w\left(f^{-1}(a)\right) \begin{cases} > d_a - y + 1 \text{ for each } a \in A_1 \\ \leq d_a + y - 1 \text{ for each } a \in A_2 \end{cases}$

To simplify the notation we introduce two further special cases of a DBE:

- If the demands are the same for each $a \in A$, e.g. $d_a = x$ for every $a \in A$ with $x \in \mathbb{N}$, then we write only the value x instead of a vector $D = \{d_a\}_{a \in A}$ in a DBE-tuple, i.e. $(A_1, A_2, y, f, D) = (A_1, A_2, y, f, x)$.

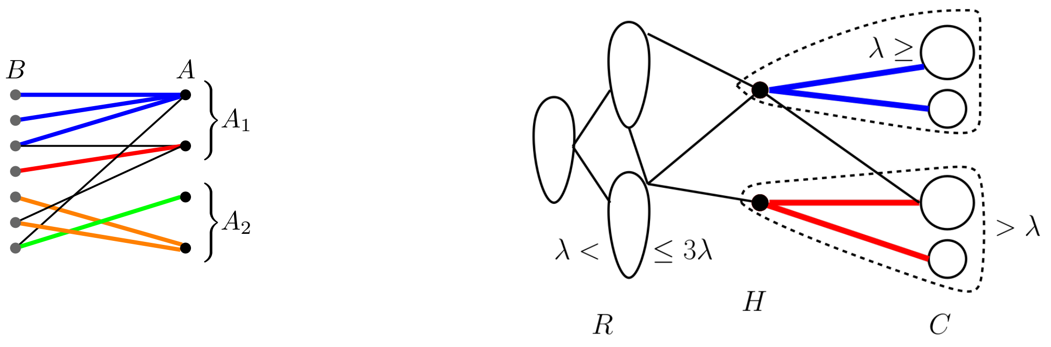

Figure 1
Left: Let $A = \{a_1, a_2, a_3, a_4\}$ be ordered in a top down manner, $w(Q) = 1$ for every $Q \in \mathbb{CC}(G[B])$ and $w(a) = 1$ for every $a \in A$. Then, $(\{a_1, a_2\}, \{a_3, a_4\}, 1, f, \{3, 1, 3, 4\})$ is a DBE, where the assignment f are depicted with corresponding colored bold edges.
Right: A λ-balanced crown decomposition, where the assignment f are depicted with corresponding colored bold edges. The two dashed lines illustrate that $w(h) + w(f^{-1}(h)) > \lambda$ for every $h \in H$ while $w(Q) \leq \lambda$ for every $Q \in \mathbb{CC}(G[C])$.

- For $q \in \mathbb{N}$ we call (C, H, f) a (q, y) crown decomposition ((q, y)-CD) if it is a $(H, \varnothing, y, f, q + y - 1)$ DBE with $H = A$ and $C = B = f^{-1}(H)$. (This term simplifies the reference to the reducible structure of crown C and head H.)

Considering a graph instance $G = (V, E)$, e.g. from VI or COC, Definition 1 usually describes only a subgraph where the reducible structure is sought, i.e. A, B are vertex subsets of V. What is crucial is that A already separates B from $V \setminus B$. Then, the vertex set $A_1 \subseteq A$ (cf. Definition 1) typically represents the structure targeted for reduction and can be seen as the "head" of a crown decomposition. The components within $\mathbb{CC}(G[B])$ assigned to A_1 (the separated part through A_1), denoted by $f^{-1}(A_1)$, form the "crown". In the context of optimization problems such as the vertex cover problem, a successful reduction often involves the head being part of an optimal solution. Subsequently, upon its removal, the crown – separated by the head – no longer needs to be considered.

Ideally, we wish to have $A_1 = A$, but even when this is not the case the balanced part of a DBE ensures that the elements mapped to A_2 are bounded, which finally allows us to bound the number of vertices (or the sum of the vertex weights) of $A_2 \cup V(f^{-1}(A_2))$.

Consider a crown decomposition, which is a partition of the vertex set into body, head and crown, where the head separates the body from the crown. The head and crown of the *classical crown decomposition* corresponds to a $(1, 1)$-CD, the *q-expansion* for $q \in \mathbb{N}$ to a $(q, 1)$-CD and the *weighted crown decomposition* for $q \in \mathbb{N}$ to a (q, x)-CD. The last known structure captured by Definition 1 is the *balanced expansion* which corresponds for $x, y \in \mathbb{N}$ to a (A_1, A_2, y, f, x) DBE. The essential new structural property of a DBE compared to a balanced expansion are the varying demands for A.

Let $G = (A \cup B, E)$ be a graph. For $A' \subseteq A$ we define $\mathcal{B}_{A'}$ as the components $Q \in \mathbb{CC}(G[B])$ with $N(Q) \subseteq A'$. The following theorem, easily derived as modification of the results of [5], gives the runtime to find a DBE and the existence guarantee of A_1 depending on the size or weight in the graph.

▶ **Theorem 2** (Demanded balanced expansion). *Let $G = (A \cup B, E, w)$ be a graph with no isolated components in $\mathbb{CC}(G[B])$, i.e. every component of $\mathbb{CC}(G[B])$ contains at least one neighbor of A. Let $y \geq \max_{Q \in \mathbb{CC}(G[B])} w(Q)$ and $D = \{d_a\}_{a \in A}$ demands with $d_a \in \mathbb{N}$ for each $a \in A$. A demanded balanced expansion (A_1, A_2, y, f, D) can be computed in $\mathcal{O}(|V|\,|E|)$ time. Furthermore, if there is an $A' \subseteq A$ with $w(A') + w(V(\mathcal{B}_{A'})) \geq \sum_{a \in A'} d_a$, then $A_1 \neq \varnothing$.*

The next graph structure that we use for our kernelization algorithms is introduced by Casel et al. [5] and is a combination of a balanced connected partition and a weighted crown decomposition, which is called *balanced crown decomposition*. Formally it is defined as follows (see also Figure 1 (right) for an illustration).

▶ **Definition 3** (λ-balanced crown decomposition)**.** *A λ-balanced crown decomposition of a graph $G = (V, E, w)$ is a tuple $(C, H, \mathfrak{R}, f)$, where $\{H, C, R\}$ is a partition of V, the set $\mathfrak{R}$ is a partition of R, and $f\colon \mathbb{CC}(G[C]) \to H$, such that:*

$$\left.\begin{array}{ll} \textbf{1.} & w(Q) \leq \lambda \text{ for each } Q \in \mathbb{CC}(G[C]), \\ \textbf{2.} & f(Q) \in N(Q) \text{ for each } Q \in \mathbb{CC}(G[C]), \\ \textbf{3.} & N(C) \subseteq H, \\ \textbf{4.} & w(h) + w(f^{-1}(h)) > \lambda \text{ for each } h \in H \text{ and} \end{array}\right\} \qquad (\lambda, \lambda)\text{-CD}$$

5. *$G[R']$ is connected and $\lambda < w(R') \leq 3\lambda$ for each $R' \in \mathfrak{R}$.*

We use λ-BCD as an abbreviation for a λ-balanced crown decomposition. Looking at the original definition of a λ-BCD in [5] (Definition 6), we shift the λ value by one, which allows us to change several inequalities between strict and non-strict for a clearer representation, while still keeping the definition the same. Furthermore, λ must be at least two in the original definition, but to simplify the understanding of the application to the problems considered in this paper, it makes more sense that a λ-BCD can also exist with $\lambda = 1$.

The authors in [5] provide an algorithm that finds a λ-BCD in polynomial time, as given in the following.

▶ **Theorem 4** (Balanced crown decomposition theorem, [5] Theorem 7)**.** *Let $G = (V, E, w)$ be a graph and $\lambda \in \mathbb{N}$, such that each connected component in G has weight larger than λ. A λ-balanced crown decomposition $(C, H, \mathfrak{R}, f)$ of G can be computed in $\mathcal{O}\left(r^2 |V| |E|\right)$ time, where $r = |H| + |\mathfrak{R}| < |V|$ is at most the size of a maximum $(\lambda + 1)$-packing.*

We end the preliminary section with a formal definition of the subgraph packing problem. Given a graph $G = (V, E)$ and two parameters $r, \lambda \in \mathbb{N}$. We say that $P_1, \ldots, P_m \subseteq V$ is a *λ-packing* if for all $i, j \in [m]$ with $i \neq j$ the induced subgraph $G[P_i]$ is connected, $|P_i| \geq \lambda$, and $P_i \cap P_j = \varnothing$. The task is to find a λ-packing of size at least r.

3 Improved Kernels for VI, wVI and wCOC

The vertex integrity of a graph models finding an optimal balance between removing vertices and keeping small connected parts of a graph. As a reminder: In the formal definition, a graph G and a parameter p are given. The task is to find a vertex set $S \subseteq V$ such that $|S| + \max_{Q \in \mathbb{CC}(G-S)} |Q|$ is at most p. In the weighted vertex integrity problem (wVI), i.e. the vertices have weights, the aim is that $w(S) + \max_{Q \in \mathbb{CC}(G-S)} w(Q)$ is at most p. We improve the vertex kernel of p^3 provided by Drange et al. [10] for both variants VI and wVI. To obtain such a kernel, they essentially established that vertices $v \in V$ with $w(v) + w(N(v)) > p$ belong to a solution if we have a yes-instance in hand. A simple counting argument after removing those vertices provides a p^3 vertex kernel. We improve this by employing crown decompositions, which offer valuable insights into the structural properties of vertex sets rather than focusing solely on individual vertices. Directly applying this method to problems like COC, as seen in prior works such as [5, 7, 19, 25], is challenging because we lack prior knowledge about the size of remaining components after solution removal. However, if we had at least a lower bound, we could theoretically safely reduce our instance accordingly. To establish such a bound, we engage in an interplay between packings and separators, where the balanced crown decomposition (BCD) proves instrumental. By identifying reducible structures in the input instance through embedding a DBE into BCD after determining a suitable bound, we prove the following theorems regarding VI and wVI.

▶ **Theorem 5.** *The vertex integrity problem admits a vertex kernel of size* $3p^2$ *in time* $\mathcal{O}\left(\log(p)|V|^4|E|\right)$.

▶ **Theorem 6.** *The weighted vertex integrity problem admits a vertex kernel of size* $3(p^2 + p^{1.5}p_\ell)$ *in time* $\mathcal{O}\left(\log(p)|V|^4|E|\right)$*, where* p_ℓ *is at most the size of the largest component after removing a solution.*

Closely related to wVI is the weighted component order connectivity problem (wCOC). Given a vertex-weighted graph $G = (V, E, w)$ and two parameters $k, W \in \mathbb{N}$, the task is to find a vertex set $S \subseteq V$ such that $w(S) \leq k$ where each component weighs at most W. The techniques employed to derive the kernel for wVI can be seamlessly applied to wCOC, thereby enhancing the current state of the art vertex kernel of $kW(k + W) + k$. This kernel is also provided by Drange et al. [10] in a similar way as for wVI.

▶ **Theorem 7.** *The weighted component order connectivity problem admits a vertex kernel of size* $3\mu(k + \sqrt{\mu}W)$*, where* $\mu = \max(k, W)$*. Furthermore, such a kernel can be computed in time* $\mathcal{O}\left(r^2k|V||E|\right)$*, where* r *is the size of a maximum* $(W + 1)$*-packing.*

Before we prove these theorems, we give some notations that we use in this section. We define them for the weighted case, as the unweighted case can be viewed in the same way with unit weights. We say that S *is a solution* if it satisfies $w(S) + \max_{Q\in\mathbb{CC}(G-S)} w(Q) \leq p$ for wIV or $w(S) \leq k$ and $\max_{Q\in\mathbb{CC}(G-S)} w(Q) \leq W$ for wCOC. We denote an instance of wIV by (G, p) and an instance of wCOC by (G, k, W). We say that (G, p) or (G, k, W) is a *yes-instance* if there is a solution, otherwise, we say that it is a *no-instance*. Let $\mathcal{S} \subseteq 2^{V(G)}$ be all solutions for (G, p). We define $p_\ell := \min_{S\in\mathcal{S}} \left(\max_{Q\in\mathbb{CC}(G-S)} w(Q)\right)$ which is the optimum lower bound on the size of the connected components after the removal of any solution; where for no-instances we set $p_\ell = p$. For all instances (G, p) of wVI, we assume $w(v) < p$ for every $v \in V$ and that G contains a connected component of weight more than p.

3.1 Vertex Integrity

The rest of this section is dedicated to the proof of Theorem 5. First, we discuss the reducible structure we are looking for and then show how to find it. Note that for $a, b \in \mathbb{N}$ with $a < b$ a (p, a)-CD is also a (p, b)-CD, but not vice versa. This means that even if we do not know p_ℓ exactly, any (p, c)-CD with $c \leq p_\ell$ is a (p, p_ℓ)-CD that can be used in the following lemma.

▶ **Lemma 8.**[1] *Let* (G, p) *be an instance of* VI and let (C, H, f) be a (p, p_ℓ)-CD in G, such that $N(C) \subseteq H$. Then, (G, p) is a yes-instance if and only if $(G - \{H \cup C\}, p - |H|)$ is a yes-instance.

In order to use Lemma 8, we must first determine a suitable lower bound for p_ℓ. Additionally, we need to ensure the existence of a corresponding weighted crown decomposition when the input graph size exceeds $3p^2$, and that we can find it efficiently. This is where the BCD comes into play. For $\lambda \in \mathbb{N}$ a λ-BCD $(C, H, \mathfrak{R}, f)$ in G can only be computed within the components of G that have a size (or weight) greater than λ. For a better readability,

[1] This result is an incorrect simplification to describe the idea. Please refer to the Full Version for the more technical correct reduction.

we will consistently assume that when computing a λ-BCD, we disregard small components, meaning that if $(C, H, \mathfrak{R}, f)$ is a λ-BCD for G, then $C \cup H \cup V(\mathfrak{R})$ are only the vertices that are contained in a component of size (or weight) more than λ of G.

We specify the following lemma directly in the weighted version to also use it later. The lemma provides a lower bound for p_ℓ by a λ-BCD $(C, H, \mathfrak{R}, f)$, where we essentially use that a λ-BCD is also a $(\lambda+1)$-packing of size $|H| + |\mathfrak{R}|$. This results from the fact that each element $R \in \mathfrak{R}$ has the size (or weight) $\lambda + 1$ and is connected and that for each $h \in H$ the subgraph $G[\{h\} \cup V(f^{-1}(h))]$ is connected and has at least the size (or weight) $\lambda + 1$. Note that the vertex sets $\{\{h\} \cup V(f^{-1}(h))\}_{h \in H} \cup \mathfrak{R}$ are pairwise disjoint.

▶ **Lemma 9.** *Let* (G, p) *be an instance of* wVI and for $\lambda \in [p]$ let $(C, H, \mathfrak{R}, f)$ be a λ-BCD in G. If $|H| + |\mathfrak{R}| > p$, then $\lambda < p_\ell$ for the instance $(G[C \cup H \cup V(\mathfrak{R})], p)$.

Note that the lower bound $\lambda + 1$ of p_ℓ in Lemma 9 is applicable to the induced graph of $V' = C \cup H \cup V(\mathfrak{R})$ within G. This implies that the isolated components of $G - V'$, defined as having a size (or weight) of at most λ, can be safely removed. Their removal does not affect the decision problem of VI or wVI concerning p. Moreover, note that an additional advantage of a $(\lambda+1)$-BCD $(C, H, \mathfrak{R}, f)$ is that the balanced part $\mathfrak{R}$ can be upper bounded by $3(\lambda+1)|\mathfrak{R}| \leq 3p_\ell|\mathfrak{R}| < 3p|\mathfrak{R}|$, while a suitable λ choice also upper bounds $|\mathfrak{R}|$ by p. In particular, if $H = C = \varnothing$ and $|\mathfrak{R}| \leq p$, then we would already have an instance with a size (or weight) of at most $3p^2$.

Clearly, a yes-instance cannot have a lower bound for p_ℓ larger than p as stated in the following corollary.

▶ **Corollary 10.** *Let* (G, p) *be an instance of* wVI. If for a p-BCD $(C, H, \mathfrak{R}, f)$ it holds that $|H| + |\mathfrak{R}| > p$, then (G, p) is a no-instance.

The next lemma shows under which conditions we can find a (p, λ)-CD in G with respect to a λ-BCD and the current graph size.

▶ **Lemma 11.** *Let* (G, p) *be an instance of* VI and for $\lambda \in [p]$ let $(C, H, \mathfrak{R}, f)$ be a λ-BCD in G with $|H| + |\mathfrak{R}| \leq p$. If $|C| + |H| + |V(\mathfrak{R})| \geq 3p^2$, then $G' = G[C \cup H]$ contains a (p, λ)-CD in G. Furthermore, we can extract it from G' in time $\mathcal{O}(|V(G')||E(G')|) \subseteq \mathcal{O}(|V(G)||E(G)|)$.

If we now combine Lemma 9 and Lemma 11, we can finally use Lemma 8 by finding a $\lambda \in [p-1]$ such that a $(\lambda+1)$-BCD $(C, H, \mathfrak{R}, f)$ satisfies $|H| + |\mathfrak{R}| \leq p$, while a λ-BCD $(C', H', \mathfrak{R}', f')$ satisfies $|H'| + |\mathfrak{R}'| > p$, which only adds a factor of $\mathcal{O}(\log(p))$ to the computation cost. Formulated more precisely: For $Q \in \mathbb{CC}(C)$ we have $|Q| \leq \lambda + 1 \leq p_\ell$ as $\lambda < p_\ell$ by Lemma 9. Further, we remove all vertices that are in components of size at most λ, which is a safe reduction rule as explained above. If the vertex size of the remaining input graph, i.e. $G[C \cup H \cup V(\mathfrak{R})]$ is at least $3p^2$, then we can find a (p, λ)-CD for G in $G[C \cup H]$ in polynomial time by Lemma 11, which is a reducible structure by Lemma 8.

With these facts in hand we can design a kernelization algorithm. It takes as input an instance (G, p) of VI and returns an equivalent instance with at most $3p^2$ vertices.

Find reducible structures (AlgVI)

1. Compute a p-BCD $(C_1, H_1, \mathfrak{R}_1, f_1)$. If $|H_1| + |\mathfrak{R}_1| > p$ return a trivial no-instance.
2. Compute a 1-BCD $(C_2, H_2, \mathfrak{R}_2, f_2)$. Let $V' = V(G) \setminus (C_2 \cup H_2 \cup V(\mathfrak{R}_2))$. If $|H_2| + |\mathfrak{R}_2| \leq p$ and $V' \neq \varnothing$, then return $(G - V', p)$.[2] If $|H_2| + |\mathfrak{R}_2| \leq p$ and $V' = \varnothing$, then compute a $(p, 1)$-CD (H_2', C_2', f_2') in $G[H_2 \cup C_2]$ and return $(G - \{H_2' \cup C_2'\}, p - |H_2'|)$.

[2] Note that in this case V' is the set of isolated vertices in G, as we compute BCD only on components of size larger than $\lambda = 1$.

3. Otherwise, for $\lambda \in [p-1]$ find a $(\lambda+1)$-BCD $(C, H, \mathfrak{R}, f)$ and a λ-BCD $(C', H', \mathfrak{R}', f')$, such that $|H| + |\mathfrak{R}| \leq p$ and $|H'| + |\mathfrak{R}'| > p$. Let $V'' = V(G) \setminus (C \cup H \cup V(\mathfrak{R}))$. If $V'' \neq \varnothing$, then return $(G - V'', p)$. Otherwise, compute a $(p, \lambda+1)$-CD (H'',C'',f'') in $G[C \cup H]$ for G and return $(G - \{C'' \cup H''\}, p - |H''|)$.

We already explained why we can safely remove V' or V'' in steps 2 and 3 if these are not empty. Note that if V' or V'' is empty, then the vertex sets of the corresponding BCD's, i.e. $C_2, H_2, V(\mathfrak{R}_2)$ and $C, H, V(\mathfrak{R})$, respectively, form a partition of $V(G)$. The correctness of step 1 is implied by Corollary 10. Step 2 is correct because $p_\ell \geq 1$ and if $V' = \varnothing$ we obtain by Lemma 11 and $|V| = |C_2| + |H_2| + |V(\mathfrak{R}_2)| \geq 3p^2$ that there must be a $(p,1)$-CD if $|H_2| + |\mathfrak{R}_2| \leq p$. Analogously, for step 3 as $p_\ell \geq \lambda + 1$ by Lemma 9 in this step. For the overall correctness of algorithm AlgVI, it remains to be proven that if we reach step 3, the required λ exists. For a λ-BCD in a binary search, we cannot ensure that $|H| + |\mathfrak{R}|$ increases with decreasing λ values, because the sizes of H and $\mathfrak{R}$ are not necessarily monotonic with respect to λ. Note, for example, that the elements in $\mathfrak{R}$ have a size range from $\lambda + 1$ to 3λ. For a successful binary search, however, it is sufficient to know that there is a λ that satisfies the desired properties for step 3. Since we only enter this step if the extreme cases (p-BCD and 1-BCD) in step 1 and 2 hold, there has to exist at least one such λ value in-between.

Finally, we prove the specified running time of Theorem 5, which completes the proof of Theorem 5. Note that for a yes-instance we have to call the algorithm AlgVI at most $|V|$ times to guarantee the desired kernel.

▶ **Lemma 12.** *Algorithm AlgVI runs in time $\mathcal{O}(\log(p)|V|^3|E|)$.*

3.2 Weighted Vertex Integrity and Component Order Connectivity

In this section, we shift our focus to the weighted variants, namely wVI and wCOC. While utilizing the packing size of the associated BCD offers a starting point for deriving a lower bound for VI, it proves insufficient for improvements in the weighted setting. Therefore, we integrate the weight of the separator H within a BCD $(C, H, \mathfrak{R}, f)$ into our analysis. Additionally, we incorporate two distinct DBE's into a BCD in case a reduction is not achievable within the respective setting. This approach enables us to establish a tighter lower bound for p_ℓ, estimate the remaining instance size more accurately to obtain the desired kernelization results (cf. Theorems 6 and 7), or identify a no-instance.

Let us start by explaining the type of reducible structure we are searching for.

▶ **Lemma 13.**[3] *Let $G = (V, E, w)$ be a vertex weighted graph, $a, b \in \mathbb{N}$, (H, H', b, f, D) a* DBE in G with $D = \{d_h\}_{h \in H}$, where $d_h = a - 2 + b \cdot (w(h) + 1)$ for each $h \in H$, and let $C = f^{-1}(H)$ with $N(C) \subseteq H$.

1. Let (G, p) be an instance of the weighted vertex integrity problem, $a = p$ and $1 \leq b \leq p_\ell$. Then, (G, p) is a yes-instance if and only if $(G - \{H \cup C\}, p - w(H))$ is a yes-instance.
2. Let (G, k, W) be an instance of the weighted component order connectivity problem and $a, b = W$. Then, (G, k, W) is a yes-instance if and only if $(G - \{H \cup C\}, k - w(H), W)$ is a yes-instance.

We are now introducing the two DBE's mentioned in conjunction with a BCD. Let $G = (V, E, w)$ be a vertex weighted graph, $a, s, \lambda \in \mathbb{N}$, $(C, H, \mathfrak{R}, f)$ a λ-BCD with $\lambda \in [a]$ and $|H| + |\mathfrak{R}| \leq s$, where $\max_{v \in V} w(v) \leq \max(a, s) =: \mu$. Let $D^Y = \{d_h^Y\}_{h \in H}$ and

[3] This result is an incorrect simplification to describe the idea. Please refer to the Full Version for the more technical correct reduction.

$D^Z = \{d_h^Z\}_{h \in H}$ be demands, where $d_h^Y = a - 2 + \lambda \cdot (w(h)+1)$ and $d_h^Z = w(h) - 1 + (\sqrt{s}+1)\lambda$ for each $h \in H$. Let $(Y_1, Y_2, \lambda, f_Y, D^Y)$ and $(Z_1, Z_2, \lambda, f_Z, D^Z)$ be DBE's in $G[C \cup H]$ with $Y_1, Y_2, Z_1, Z_2 \subseteq H$ and $f_Y^{-1}(Y_1), f_Y^{-1}(Y_1), f_Z^{-1}(Z_1), f_Z^{-1}(Z_2) \subseteq \mathbb{CC}(C)$. Observe that if $Y_1 \neq \varnothing$ (resp. $Z_1 \neq \varnothing$) then this set separates $f_Y^{-1}(Y_1)$ (res. $f_Z^{-1}(Z_1)$) from the rest of the graph, since H separates C from the rest of the graph. Thus, if $a = p$ and $\lambda \leq p_\ell$ considering wVI or $a, \lambda = W$ considering wCOC and $Y_1 \neq \varnothing$, then in both cases we would have a reducible structure for the corresponding problem (cf. Lemma 13). We conclude this section by presenting two crucial lemmas. These lemmas serve as the foundation for designing the algorithms needed to prove Theorems 6 and 7. The first lemma aims to estimate the instance size, while the second lemma helps establish a more precise lower bound for p_ℓ or identify instances with no feasible solutions.

▶ **Lemma 14.** *Let $w(Z_1) \leq 2\mu^{1.5}$. If $Y_1 = \varnothing$, then $w(V) < 3\mu(s + \sqrt{\mu}\lambda)$.*

▶ **Lemma 15.** *If $w(Z_1) > 2\mu^{1.5}$, then there is no separator S such that $w(S) \leq s$ and $\max_{Q \in \mathbb{CC}(G-S)} w(Q) \leq \lambda$.*

4 Kernels for Component Order Connectivity

In this section, we consider COC, a refined version of VI. Given a graph $G = (V, E)$ and parameters $k, W \in \mathbb{N}$, COC aims to remove at most k vertices so that resulting components have at most W vertices each. We present a kernelization algorithm that provides a $2kW$ vertex-kernel in an FPT-runtime when parameterized by the size of a maximum $(W+1)$-packing, as stated in the following theorem.

▶ **Theorem 16.** *A vertex kernel of size $2kW$ for the component order connectivity problem can be computed in time $\mathcal{O}(r^3|V||E| \cdot r^{\min(3r,k)})$, where $r \leq k$ is the size of a maximum $(W+1)$-packing.*

A kernel of size $2kW$ is also presented in [19], however with an FPT-runtime in the parameter W and using linear programming methods. In contrast, our result has an FPT-runtime in the parameter of a maximum $(W+1)$-packing $r \leq k$ and is fully combinatorial. Moreover, we showcase how our algorithm transforms into a polynomial one for two cases: when $W = 1$ (Vertex Cover) and for claw-free graphs. While generalizing our FPT-runtime algorithm into polynomial time eludes us, our understanding of crown decomposition's structural properties holds promise for future progress.

For VI, wVI, and wCOC, crown structures were integrated into the head and crown of the BCD. In contrast, we now incorporate them into the body of the BCD. Of particular interest are the so-called strictly reducible pairs introduced by Kumar and Lokshtanov [19] who prove that such a structure must exist in graphs with more than $2kW$ vertices.

▶ **Definition 17** ((strictly) reducible pair)**.** *For a graph $G = (V, E)$ and a parameter $W \in \mathbb{N}$, a pair (A, B) of vertex disjoint subsets of V is a* reducible pair *for* COC if the following conditions are satisfied:

- $N(B) \subseteq A$.
- The size of each $Q \in \mathbb{CC}(G[B])$ is at most W.
- There is an assignment function $g \colon \mathbb{CC}(G[B]) \times A \to \mathbb{N}_0$, such that
  - for all $Q \in \mathbb{CC}(G[B])$ and $a \in A$, if $g(Q, a) \neq 0$, then $a \in N(Q)$
  - for all $a \in A$ we have $\sum_{Q \in \mathbb{CC}(G[B])} g(Q, a) \geq 2W - 1$,
  - for all $Q \in \mathbb{CC}(G[B])$ we have $\sum_{a \in A} g(Q, a) \leq |Q|$.

In addition, if there exists an $a \in A$ such that $\sum_{Q \in \mathbb{CC}(G[B])} g(Q, a) \geq 2W$, then (A, B) is a *strictly reducible pair.*

We say (A, B) is a minimal (strictly) reducible pair if there does not exist a (strictly) reducible pair (A', B') with $A' \subset A$ and $B' \subseteq B$. A (strictly) reducible pair is basically a weighted crown decomposition where A is the head and B is the crown.

In essence, the kernelization algorithm outlined below focuses on analyzing pairs (A, B) in a W-BCD. We show that the head vertices A have specific traits in a W-BCD, making it possible to locate them. Structurally, the algorithm operates as a bounded search tree, hence it is presented recursively. It takes an instance (G, k, W) of COC as input, where $|V(G)| > 2kW$ and each component of G has at least $W + 1$ vertices. The size limits for the input graph are not arbitrary; if $|V(G)| \leq 2kW$, there is nothing to do, and removing components smaller than $W + 1$ is a safe reduction.

For $V' \subseteq V$ we define $\texttt{sep}_W(V') \in \mathbb{N}$ as the cardinality of a minimum W-separator in $G[V']$, and $\texttt{arg sep}_W(V') \subset V$ as an argument suitable to this cardinality. For a graph $G' \subseteq G$ let $G^{>W}(G')$ be the graph obtained by removing all components of size at most W from G. Lastly, for a W-BCD $(C, H, \mathfrak{R}, f)$ we define $\mathfrak{R}' := \{R \in \mathfrak{R} \mid |R| > 2W \text{ and } \texttt{sep}_W(R) = 1\}$ and $S_{\mathfrak{R}'} := \bigcup_{R \in \mathfrak{R}'} \texttt{arg sep}_W(R)$. (uniqueness of $\texttt{arg sep}_W(R)$ for $R \in \mathfrak{R}'$ is shown in the full version [6]).

Find reducible structures (AlgCOC)

```
1. Compute a $W$-BCD $(C, H, \mathfrak{R}, f)$ in $G$ and initialize $t = |H| + |\mathfrak{R}|$ and $S = \varnothing$.
2. Let $G' = G^{>W}(G - S)$. If $G'$ is an empty graph return a trivial yes-instance. Otherwise,
   compute a $W$-BCD $(C, H, \mathfrak{R}, f)$ in $G'$.
   a. If $|H| + |\mathfrak{R}| > k$ return a trivial no-instance.
   b. Let $\mathcal{Q}$ be the connected components of size at most $W$ in $G - S$. Let $A = S \cup H$ and
      $B = C \cup V(\mathcal{Q})$. Compute a DBE $(A_1, A_2, W, f, 2W - 1)$ in $G[A \cup B]$ by using Theorem 2.
      If $A_1 \neq \varnothing$, then terminate the algorithm and return $(G - (A_1 \cup f^{-1}(A_1)), k - |A_1|, W)$.
3. If the depth of the recursion is more than $\min(3t, k)$, then break.
4. For each $v \in H \cup S_{\mathfrak{R}'}$:
   - Add $v$ to $S$ and recurse from step 2.
5. Return a trivial no-instance.
```

The interesting part of the algorithm is the localization of a minimal strictly reducible pair if it exists in the graph. Let (A, B) be a minimal strictly reducible pair in G. As already mentioned, algorithm AlgCOC is basically a bounded search tree with a working vertex set S, which are potential head vertices. The crucial step is ensuring that S consistently matches the vertex set A. Once this alignment is achieved, localization of (A, B) is assured, as step 2b (specifically Theorem 2) guarantees its extraction.

We conclude this part of the section by presenting a crucial lemma along with its implication for a depth-wise progression within the bounded search tree towards $S = A$ in algorithm AlgCOC. Let (A, B) be a minimal strictly reducible pair in $G = (V, E)$ and let $S \subset V$. Let $(C, H, \mathfrak{R}, f)$ be a W-BCD in $G^{>W}(G - S)$.

▶ **Lemma 18.** *If $S \cap B = \varnothing$ and $S \nsubseteq A$, then $(H \cup S_{\mathfrak{R}'}) \cap A \neq \varnothing$.*

If we reach step 4, we have no reducible pair in hand so far and can therefore assume that $S \neq A$. Thus Lemma 18 provides the following relation to the algorithm: if $S \subset A$ (where S is possibly empty), then we can find at least one vertex of A in $H \cup S_{\mathfrak{R}'}$. Since the algorithm considers expanding S for each vertex of $H \cup S_{\mathfrak{R}'}$, at least one of which comes from A, and repeats this process with the resulting vertex set, we finally arrive at the case $S = A$.

Next, we introduce another algorithm designed to compute a $2kW$ vertex kernel for COC. This algorithm shares the same principles as AlgCOC but exhibits polynomial running times for two specific cases.

Find reducible structures (AlgCOC-2)

1. Initialize $S = \varnothing$ and $G' = G$.
2. While $G' \neq (\varnothing, \varnothing)$:
   a. Compute a W-BCD $(C, H, \mathfrak{R}, f)$ in G', such that for a minimal strictly reducible pair (A, B) in G we have $B \cap S_{\mathfrak{R}'} = \varnothing$.
   b. If $|H| + |\mathfrak{R}| > k$, or $\mathfrak{R}' = \varnothing$ and $H = \varnothing$, then terminate the while-loop.
   c. Add the vertices $S_{\mathfrak{R}'}$ and H to S.
   d. Let $\mathcal{Q}$ be the connected components of size at most W in $G - S$. Let $A = S \cup H$, $B = C \cup V(\mathcal{Q})$. Compute a DBE $(A_1, A_2, W, f, 2W-1)$ in $G[A \cup B]$ by using Theorem 2. If $A_1 \neq \varnothing$, then terminate the algorithm and return $(G - (A_1 \cup f^{-1}(A_1)), k - |A_1|, W)$.
   e. Update G' by $G^{>W}(G - S)$.
3. Return a trivial no-instance.

The correctness proof of AlgCOC-2 can be found in the extended version [6]. The crucial difference to the previous algorithm is step 2a. Unfortunately, we have not succeeded in finding a polynomial algorithm in general for this step.

To introduce a novel approach for computing a $2k$ vertex kernel for the vertex cover problem, we present a result that establishes a $2kW$ vertex kernel for COC, applicable for any given W. However, the corresponding algorithm achieves polynomial runtime only when $W = 1$. This limitation arises because computing a maximum $(W + 1)$-packing is polynomially solvable only for $W = 1$ where it corresponds to a maximum matching. To understand how we can apply algorithm AlgCOC-2 (basically ensuring $B \cap S_{\mathfrak{R}'} = \varnothing$ in any iteration of the while-loop) we need to extend the algorithm to compute a W-BCD $(C, H, \mathfrak{R}, f)$. First observe that $\mathcal{P} = \mathfrak{R} \cup \{\{h\} \cup V(f^{-1})(h)\}_{h \in H}$ forms a $(W + 1)$-packing of size $|H| + |\mathfrak{R}|$. In order to find $\mathfrak{R}$ and H in the algorithm computing a W-BCD there are two according working sets, $\mathfrak{R}'$ and H', that always guarantee a $(W + 1)$-packing of size $|H'| + |\mathfrak{R}'|$. In particular, these sets measure the progress in a sense that the corresponding packing can only increase. The crucial point now is that we can start the algorithm with an arbitrary maximal $(W+1)$-packing and if this is a maximum $(W+1)$-packing, then after the termination of the algorithm, $\mathcal{P}$ corresponds also to a maximum $(W + 1)$-packing. In fact, the original algorithm initializes $\mathfrak{R}'$ in the beginning as the connected components of the input graph and $H' = \varnothing$, but it is also possible to initialize $\mathfrak{R}'$ as a maximal $(W+1)$-packing instead (see [5], proof sketch of Theorem 7). The relationship between a W-BCD, where $\mathcal{P}$ represents a maximum $(W + 1)$-packing, and a reducible pair (A, B) yields a significant property: every element in $\mathcal{P}$ intersects with at most one vertex of A. However, if a vertex of B is in $S_{\mathfrak{R}'}$ in algorithm AlgCOC-2, then the according $R \in \mathfrak{R}'$ must contain at least two vertices of A. This contradicts the aforementioned property. These findings form the key point of the following theorem.

▶ **Theorem 19.** *For $W = 1$, i.e. for the vertex cover problem, algorithm* AlgCOC-2 works correctly and provides a $2k$ vertex kernel in polynomial time.

We also demonstrate the application of AlgCOC-2 for claw-free graphs, i.e., graphs without induced $K_{1,3}$'s. To achieve this, we use a vertex partitioning algorithm tailored for such graphs, as proposed by Borndörfer et al. [4]. Essentially, we establish that claw-free graphs can be represented by a W-BCD $(C, H, \mathfrak{R}, f)$ with C and H being empty, and at most one element of $\mathfrak{R}$ exceeding size $2W$. Utilizing this along with Lemma 18 ensures that $B \cap S_{\mathfrak{R}'} = \varnothing$ in any iteration of the while-loop and provides basically the following theorem.

▶ **Theorem 20.** *The component order connectivity problem admits a $2kW$ vertex-kernel on claw-free graphs in polynomial time.*

References

1 Kunwarjit S. Bagga, Lowell W. Beineke, Wayne Goddard, Marc J. Lipman, and Raymond E. Pippert. A survey of integrity. *Discret. Appl. Math.*, 37/38:13–28, 1992. doi:10.1016/0166-218X(92)90122-Q.

2 Curtis A Barefoot, Roger Entringer, and Henda Swart. Vulnerability in graphs-a comparative survey. *J. Combin. Math. Combin. Comput*, 1(38):13–22, 1987.

3 Matthias Bentert, Klaus Heeger, and Tomohiro Koana. Fully polynomial-time algorithms parameterized by vertex integrity using fast matrix multiplication. In Inge Li Gørtz, Martin Farach-Colton, Simon J. Puglisi, and Grzegorz Herman, editors, *31st Annual European Symposium on Algorithms, ESA 2023, September 4-6, 2023, Amsterdam, The Netherlands*, volume 274 of *LIPIcs*, pages 16:1–16:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPICS.ESA.2023.16.

4 Ralf Borndörfer, Katrin Casel, Davis Issac, Aikaterini Niklanovits, Stephan Schwartz, and Ziena Zeif. Connected k-partition of k-connected graphs and c-claw-free graphs. In Mary Wootters and Laura Sanità, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2021, August 16-18, 2021, University of Washington, Seattle, Washington, USA (Virtual Conference)*, volume 207 of *LIPIcs*, pages 27:1–27:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPICS.APPROX/RANDOM.2021.27.

5 Katrin Casel, Tobias Friedrich, Davis Issac, Aikaterini Niklanovits, and Ziena Zeif. Balanced crown decomposition for connectivity constraints. In *29th Annual European Symposium on Algorithms, ESA 2021, September 6-8, 2021, Lisbon, Portugal (Virtual Conference)*, volume 204 of *LIPIcs*, pages 26:1–26:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.ESA.2021.26.

6 Katrin Casel, Tobias Friedrich, Aikaterini Niklanovits, Kirill Simonov, and Ziena Zeif. Combining crown structures for vulnerability measures. *CoRR*, abs/2405.02378, 2024. doi:10.48550/arXiv.2405.02378.

7 Jianer Chen, Henning Fernau, Peter Shaw, Jianxin Wang, and Zhibiao Yang. Kernels for packing and covering problems. *Theor. Comput. Sci.*, 790:152–166, 2019. doi:10.1016/J.TCS.2019.04.018.

8 Lane H Clark, Roger C Entringer, and Michael R Fellows. Computational complexity of integrity. *J. Combin. Math. Combin. Comput*, 2:179–191, 1987.

9 Rodney G Downey and Michael Ralph Fellows. *Parameterized complexity.* Springer Science & Business Media, 2012. doi:10.1007/978-1-4612-0515-9.

10 Pål Grønås Drange, Markus S. Dregi, and Pim van 't Hof. On the computational complexity of vertex integrity and component order connectivity. *Algorithmica*, 76(4):1181–1202, 2016. doi:10.1007/s00453-016-0127-x.

11 Michael R Fellows and Sam Stueckle. The immersion order, forbidden subgraphs and the complexity of network integrity. *J. Combin. Math. Combin. Comput*, 6(1):23–32, 1989.

12 Fedor V Fomin, Daniel Lokshtanov, Saket Saurabh, and Meirav Zehavi. *Kernelization: theory of parameterized preprocessing.* Cambridge University Press, 2019. doi:10.1017/9781107415157.

13 Tatsuya Gima, Tesshu Hanaka, Masashi Kiyomi, Yasuaki Kobayashi, and Yota Otachi. Exploring the gap between treedepth and vertex cover through vertex integrity. *Theor. Comput. Sci.*, 918:60–76, 2022. doi:10.1016/J.TCS.2022.03.021.

14 Tatsuya Gima, Tesshu Hanaka, Yasuaki Kobayashi, Ryota Murai, Hirotaka Ono, and Yota Otachi. Structural parameterizations of vertex integrity. In Ryuhei Uehara, Katsuhisa Yamanaka, and Hsu-Chun Yen, editors, *WALCOM: Algorithms and Computation - 18th International Conference and Workshops on Algorithms and Computation, WALCOM 2024, Kanazawa, Japan, March 18-20, 2024, Proceedings*, volume 14549 of *Lecture Notes in Computer Science*, pages 406–420. Springer, 2024. doi:10.1007/978-981-97-0566-5_29.

15 Tatsuya Gima and Yota Otachi. Extended MSO model checking via small vertex integrity. *Algorithmica*, 86(1):147–170, 2024. doi:10.1007/S00453-023-01161-9.

16 Tesshu Hanaka, Michael Lampis, Manolis Vasilakis, and Kanae Yoshiwatari. Parameterized vertex integrity revisited. *CoRR*, abs/2402.09971, 2024. doi:10.48550/arXiv.2402.09971.

17 Ashwin Jacob, Diptapriyo Majumdar, and Venkatesh Raman. Expansion lemma - variations and applications to polynomial-time preprocessing. *Algorithms*, 16(3):144, 2023. doi:10.3390/A16030144.

18 Dieter Kratsch, Ton Kloks, and Haiko Müller. Measuring the vulnerability for classes of intersection graphs. *Discret. Appl. Math.*, 77(3):259–270, 1997. doi:10.1016/S0166-218X(96)00133-3.

19 Mithilesh Kumar and Daniel Lokshtanov. A 2lk kernel for l-component order connectivity. In Jiong Guo and Danny Hermelin, editors, *11th International Symposium on Parameterized and Exact Computation, IPEC 2016, August 24-26, 2016, Aarhus, Denmark*, volume 63 of *LIPIcs*, pages 20:1–20:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPIcs.IPEC.2016.20.

20 Michael Lampis and Valia Mitsou. Fine-grained meta-theorems for vertex integrity. In Hee-Kap Ahn and Kunihiko Sadakane, editors, *32nd International Symposium on Algorithms and Computation, ISAAC 2021, December 6-8, 2021, Fukuoka, Japan*, volume 212 of *LIPIcs*, pages 34:1–34:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPICS.ISAAC.2021.34.

21 Euiwoong Lee. Partitioning a graph into small pieces with applications to path transversal. *Math. Program.*, 177(1-2):1–19, 2019. doi:10.1007/s10107-018-1255-7.

22 Wenjun Li and Binhai Zhu. A $2k$-kernelization algorithm for vertex cover based on crown decomposition. *Theor. Comput. Sci.*, 739:80–85, 2018. doi:10.1016/J.TCS.2018.05.004.

23 Yinkui Li, Shenggui Zhang, and Qilong Zhang. Vulnerability parameters of split graphs. *Int. J. Comput. Math.*, 85(1):19–23, 2008. doi:10.1080/00207160701365721.

24 Mingyu Xiao. Linear kernels for separating a graph into components of bounded size. *J. Comput. Syst. Sci.*, 88:260–270, 2017. doi:10.1016/J.JCSS.2017.04.004.

25 Mingyu Xiao and Shaowei Kou. Kernelization and parameterized algorithms for 3-path vertex cover. In T. V. Gopal, Gerhard Jäger, and Silvia Steila, editors, *Theory and Applications of Models of Computation - 14th Annual Conference, TAMC 2017, Bern, Switzerland, April 20-22, 2017, Proceedings*, volume 10185 of *Lecture Notes in Computer Science*, pages 654–668, 2017. doi:10.1007/978-3-319-55911-7_47.