Balanced Crown Decomposition for Connectivity Constraints

Katrin Casel
Hasso Plattner Institute, University of Potsdam, Germany

Tobias Friedrich
Hasso Plattner Institute, University of Potsdam, Germany

Davis Issac
Hasso Plattner Institute, University of Potsdam, Germany

Aikaterini Niklanovits
Hasso Plattner Institute, University of Potsdam, Germany

Ziena Zeif
Hasso Plattner Institute, University of Potsdam, Germany

Abstract

We introduce the *balanced crown decomposition* that captures the structure imposed on graphs by their connected induced subgraphs of a given size. Such subgraphs are a popular modeling tool in various application areas, where the non-local nature of the connectivity condition usually results in very challenging algorithmic tasks. The balanced crown decomposition is a combination of a crown decomposition and a balanced partition which makes it applicable to graph editing as well as graph packing and partitioning problems. We illustrate this by deriving improved approximation algorithms and kernelization for a variety of such problems.

In particular, through this structure, we obtain the first constant-factor approximation for the Balanced Connected Partition (BCP) problem, where the task is to partition a vertex-weighted graph into k connected components of approximately equal weight. We derive a 3-approximation for the two most commonly used objectives of maximizing the weight of the lightest component or minimizing the weight of the heaviest component.

2012 ACM Subject Classification Theory of computation → Graph algorithms analysis

Keywords and phrases crown decomposition, connected partition, balanced partition, approximation algorithms

Digital Object Identifier 10.4230/LIPIcs.ESA.2021.26

Related Version *Full Version*: `https://arxiv.org/abs/2011.04528`

1 Introduction

Connected subgraphs are one of the most natural structures to encode aspects of a practical task, modeled as a graph problem. On the one side, such subgraphs represent structures we seek to discover, such as territories for postal delivery and similar districting problems (see e.g. the survey [22]). From another perspective, the structures of interest could be operations that scatter a graph into small connected components; a structure e.g. used to model vulnerability in network security (see e.g. the survey [2]). Partitioning a graph into connected components of a given size is also used as a model for task allocation to robots [43]. From an algorithmic perspective, connectivity is a non-local requirement which makes it particularly challenging. We introduce a graph structure that can be used to design efficient algorithms for a broad class of problems involving connected subgraphs of a given size.

One useful structure to derive information about connected subgraphs is a *connected partition*, defined as follows. Given a graph $G = (V, E)$, a connected partition of G is a partition $V_1, \ldots, V_k$ of V such that the graph induced by the vertex set V_i is connected for each $1 \leq i \leq k$, where $k \in \mathbb{N}$ is the size of the partition. Often, we are not interested in just any connected partition but in those that have the additional property of being *balanced*. Informally speaking, a connected partition is considered balanced if the sets V_i have approximately equal cardinality. There are several measures to assess the quality of a *balanced connected partition* (BCP for short), with the two most commonly used objectives being to maximize $\min_{1 \leq i \leq k} |V_i|$ or to minimize $\max_{1 \leq i \leq k} |V_i|$, as first introduced for trees in [30] and [25], respectively. Despite extensive studies on these problems for the past 40 years, see e.g. [7, 30, 32, 35, 37, 38, 39] the best known approximation ratio for the Min-Max objective depends on the number of sets k [11]. For the Max-Min objective not even such a result is known; only for the cases with k restricted to 2 or 3, there exist approximations with ratio $\frac{4}{3}$ [12] and $\frac{5}{3}$ [8], respectively. Deriving an approximation with a ratio independent of k seems to require a new strategy.

Helpful structures for both of the objectives Max-Min and Min-Max, as a sort of compromise, are BCP's such that $\lambda_l \leq |V_i| \leq \lambda_r$ for some fixed bounds λ_l, λ_r. We call this compromise structure $[\lambda_l, \lambda_r]$-CVP (connected *vertex* partition), and it is one ingredient of *balanced crown decomposition*, the main structural object that we present. In the case that no $[\lambda_l, \lambda_r]$-CVP exists for a graph, we can learn something about its structure. In particular, our *balanced crown decomposition theorem* (Theorem 7) shows that for any $k, \lambda > 0$, the non-existence of a $[\lambda, 3\lambda - 3]$-CVP implies the existence of a vertex set $H \subseteq V$ of cardinality at most k that disconnects at least one component of size less than λ from G. Such sets H are in a sense the dual of balanced connected partitions.

Small subsets of vertices that disconnect a graph are usually called *vertex separators*, and they are one of the most powerful tools for designing efficient graph algorithms. In a sense, they are the base requirement of successful divide-and-conquer strategies for graph problems. This generality and their wide applicability has made the study of separators a rich and active research field, see e.g. the book by Rosenberg and Heath [31], or the line of research initialized by the seminal paper of Lipton and Tarjan [27] on separators in planar graphs. Numerous different types of separator structures emerged over the past couple of decades. In the context of connectivity problems, the separator structures of particular interest are crown decompositions; a classical tool to derive kernelizations in the field of parameterized complexity. We refer to chapter 4 of the book on kernelization [18] for more details on crown decompositions and their applications.

Crown decomposition was introduced as a generalization of Hall's Theorem in [13]. More precisely, a *crown decomposition* of a graph $G = (V, E)$ is a partition of V into three sets H (*head*), C (*crown*) and R (*royal body*), such that H separates C from R, C is an independent set in G, and there exists a matching of size $|H|$ among the edges $E \cap (H \times C)$. Notice that the set H is the separator set, and the property of C being an independent set can be seen as H splitting connected components of size 1 from the graph. The condition of the matching from H into C models a trade-off between the size of the separator and the amount of small sets that are separated. Different versions of crown decompositions have been introduced in the literature, adjusting the structure to specific application scenarios.

The structure of particular interest to us is the *q-Weighted Crown Decomposition* introduced by Xiao [40]. Here, the crown C is no longer an independent set, but has the restriction that each connected component in it has size at most q (generalizing the notion of independent set for $q = 1$); and there exist an assignment of connected components of

the crown to the head such that each vertex in the head is assigned at least q vertices. This assignment generalizes the notion of matching in the original crown decomposition. Such a weighted crown decomposition can be derived using the *Expansion Lemma* as stated in [17, Chapter 5.3], and its generalization to the *Weighted Expansion Lemma* as given in slightly different forms in [24] and [40]. The *expansions* derived by these lemmas can be thought of as bipartite analogues of the crown decomposition. Formally, given a bipartite graph $G = (A, B, E)$, a q-(Weighted) Expansion is given by sets $H \subseteq A$ and $C \subseteq B$ such that the neighborhood of C is contained in H and an assignment $f\colon C \to H$ such that the number (resp. weight, in the vertex-weighted case) of vertices assigned to each vertex in H is at least q (resp. $q - W + 1$ where W is the largest weight). Both Kumar and Lokshtanov [24] and Xiao [40] use their respective Weighted Expansion Lemma to derive kernels for Component Order Connectivity, a version of the editing problem that we also consider in a more general form under the name W-Weight Separator.

To create our new structure *balanced crown decomposition*, we combine balanced connected partitions and crown decompositions to derive a tool that has the advantages of both of the individual structures. Essentially, it is a weighted crown decomposition with the additional property that the body has a balanced connected partition. Also, note that we allow a more generalized version of weighted crown decomposition than Xiao [40], by considering weighted vertices. Formally, we consider *vertex-weighted* graphs $G = (V, E, w)$ with integer-weights $w\colon V \to \mathbb{N}$. For simplicity we use $w(V') = \sum_{v \in V'} w(v)$ for the *weight of a subset* $V' \subseteq V$.

We show that balanced crown decompositions have applications for various kinds of problems involving connectivity constraints. Specifically we discuss for the three types editing, packing and partitioning the following problems on input $G = (V, E, w)$ and $k, W \in \mathbb{N}$:

- Max-Min (Min-Max) BCP: Decide if there exists a connected partition $V_1, \ldots, V_k$ of V such that $w(V_i) \geq W$ (resp. $w(V_i) \leq W$) for each $i \in [k]$; usually stated as optimization problem to maximize/minimize W.
- W-weight Separator: Decide if there exists a set $S \subseteq V$ with $|S| \leq k$ whose removal from G yields a graph with each connected component having weight less than W.
- W-weight Packing: Decide if there exist k pairwise disjoint sets $V_1, \ldots, V_k \subseteq V$ with $w(V_i) \geq W$, such that the graph induced by V_i is connected, for each $i \in [k]$.

We remark that the problems W-weight Separator and W-weight Packing have been studied mostly on the unweighted versions, also known as Component Order Connectivity and T_r-Packing, respectively.

For all results of this paper, we consider the RAM model of computation with word size $O(\log(|V| + \max_{v \in V} w(v)))$. All our algorithms are polynomial w.r.t. the encoding of input.

Lastly, we point out that this short version of the paper contains only a summary of how to compute a balanced crown decomposition and its applications. Here we provide descriptions of the algorithms used and proof sketches about the results achieved through this structure. For technical details and complete proofs, we refer to the full version of the paper.

1.1 Our Contribution

Our main contributions can be summarized as follows:

1. **Balanced Crown Decomposition** (BCD): The main contribution of our paper is a new crown decomposition tailored for problems with connectivity constraints. Our novel addition over previous crown decompositions is that we also give a partition of the *body* into connected parts of roughly similar size. More precisely, we divide the graph into

C, H, and R such that C, H, R is a weighted-crown decomposition and also a $[\lambda, 3\lambda]$-CVP is given for R. Definition 6 gives the formal definition of BCD, and Theorem 7 gives our main result about computing BCD. We believe that apart from the applications used in this paper, BCD will find applications for other problems with connectivity constraints.

2. **Balanced Expansion**: We also give a novel variation of the expansion lemma, which is an important constituent of our algorithm for BCD. Given a bipartite graph $G = (A, B, E)$, we give an expansion with $H \subseteq A, C \subseteq B$, with the addition that the expansion f while being a weighted q-expansion from C to H, now also maps $B \setminus C$ to neighbors in $A \setminus H$ such that only a bounded weight is assigned to each vertex in $A \setminus H$. See Definition 1 for a more formal definition and Theorem 2 for our result on computing balanced expansion. Apart from its usage here to compute BCD, the balanced expansion could be of independent interest, given the significance of the Expansion Lemma in parameterized complexity.
3. **Approximation algorithms for BCP**: Using BCD, we give 3-approximation algorithms for both Max-Min and Min-Max BCP. These are the first constant approximations for both problems in polynomial time for a general k. Recall that despite numerous efforts in the past 40 years, only a $k/2$-approximation for Min-Max BCP [11], and constant-factor approximations for the particular cases of $k = 2, 3$ for Max-Min BCP [12, 8] were known.
4. **Improved Kernels for W-weight separator and packing**: BCD directly gives a $3kW$-kernel for both of the problems improving over the previous best polynomial time kernels of size $9kW$ [40] and $\mathcal{O}(kW^3)$ [9]. Especially, we get the same improvements for the unweighted versions Component Order Connectivity and T_r-Packing.
5. **Faster algorithms for Expansion**: Our algorithm for Balanced Expansion, also gives an alternative flow-based method for computing the standard (weighted) expansion. Our algorithm can compute a (weighted) expansion in $\mathcal{O}(|V||E|)$ surpassing the previous best runtimes of $\mathcal{O}(|V|^{1.5}|E|)$ and $\mathcal{O}(|E||V|^{1.5}W^{2.5})$ [18, Chapter 5.3] (here W is the largest weight) for unweighted and weighted expansion, respectively. In particular, for weighted expansion, our runtime does not depend on the weights and is the first algorithm that runs in time polynomial w.r.t. the length of the input-encoding. The improvement in runtime may turn out to be useful to speed up kernelizations for other problems.

1.2 Related work

Both variants of BCP were first introduced for trees, where Max-Min BCP and Min-Max BCP are introduced in [30] and [25], respectively. For this restriction to trees, a linear time algorithm was provided for both variants in [19]. For both variants of BCP, a Δ_T-approximation is given in [3] where Δ_T is the maximal degree of an arbitrary spanning tree of the input graph; for Max-Min BCP the result holds only when the input is restricted to weights with $\max_{v \in V} w(v) \leq \frac{w(G)}{\Delta k}$. Also, although not explicitly stated, a $(1 + ln\,(k))$-approximation in $\mathcal{O}\left(n^k\right)$ time for Min-Max BCP$_k$ follows from the results in [10]. With respect to lower bounds, it is known that there exists no approximation for Max-Min BCP with a ratio below 6/5, unless $\mathsf{P} \neq \mathsf{NP}$ [7]. For the unweighted case, i.e. $w \equiv 1$, the best known result for Min-Max BCP is the $\frac{k}{2}$-approximation for every $k \geq 3$ given in [11].

Balanced connected partitions are also studied for particular cases of k, denoted BCP$_k$. The restriction BCP$_2$, i.e. balanced connected bipartition, is already be **NP**-hard [5]. On the positive side, a $\frac{4}{3}$-approximation for Max-Min BCP$_2$ is given in [12], and in [11] this result is used to derive a $\frac{5}{4}$-approximation for Min-Max BCP$_2$. For tripartition, approximations for Max-Min BCP$_3$ and Min-Max BCP$_3$ with ratios $\frac{5}{3}$ and $\frac{3}{2}$, respectively, are given in [8].

BCP in unit-weighted k-connected graphs can be seen as a special case of the Győri-Lovász Theorem (independently given by Győri [20] and Lovász [28]). It states that for any k-connected graph $G = (V, E)$ and integers $n_1, \ldots, n_k$ with $n_1 + \cdots + n_k = |V|$, there exists

a connected partition $V_1, \dots, V_k$ of V with $|V_i| = n_i$ for all $i \in [k]$. Moreover, it is possible to fix vertices $v_1, \dots, v_k$ and request $v_i \in V_i$ for all $i \in [k]$. The Győri-Lovász Theorem is extended to weighted directed graphs in [10] and Győri's original proof is generalized to weighted undirected graphs in [6]. Polynomial algorithms to also compute such connected partitions are only known for the particular cases $k = 2, 3, 4$ [32, 35, 21] and all $k \geq 5$ are still open. A restricted case of BCP where the partitions are allowed to differ only by a size of one, has been studied from the FPT viewpoint [15].

W-WEIGHT SEPARATOR occurs in the literature under different names. The unweighted version is studied under the names *p-Size Separator* [40] or *ℓ-Component Order Connectivity* (COC) [14, 24]; where $p, \ell = W - 1$ translate this to our definition of W-WEIGHT SEPARATOR with unit weights. In [14] a weighted version of COC denoted by *Weighted Component Order Connectivity* (wCOC) is introduced. This problem differs from our W-WEIGHT SEPARATOR by searching for a set S with $w(S) \leq k$ instead of $|S| \leq k$.

Note that W-WEIGHT SEPARATOR with $W = 2$ and unit weights yields the classical problem VERTEX COVER. This in particular shows that W (alone) is not a suitable parameter from the FPT viewpoint. Further, W-WEIGHT SEPARATOR is **W[1]**-hard for parameter k, even when restricted to split graphs [14]. These lower bounds lead to studying parameterization by $W + k$. Stated with $\ell = W - 1$, a kernel of size $9k\ell$ is given in [40]. Also [24] derives a kernel of size $2k\ell$ in time $\mathcal{O}\left(|V|^{\ell}\right)$. Both of these results are for unit weights. An $\mathcal{O}\left(k\ell(k+\ell)^2\right)$ weight kernel for the related problem w-COC is given in [14].

For $W = 3$ and unit weights, W-WEIGHT SEPARATOR corresponds to VERTEX COVER P_3 or 3-PATH VERTEX COVER (see e.g. [34] and [4]), first studied by Yannakakis [42] under the name DISSOCIATION NUMBER. The best known kernel for this problem is of size $5k$ and given in [41]. W-WEIGHT PACKING with unit weights is equivalent to T_r-PACKING with $r = W + 1$, where T_r is a tree with at least r edges, as defined in [9]; note that any connected component with at least W vertices has at least $r - 1$ edges, and any tree with $r - 1$ edges has exactly W vertices. The best known kernel for this problem is of size $\mathcal{O}\left(kW^3\right)$ by [9].

W-WEIGHT PACKING is also studied for particular values of W. The case $W = 2$ with unit weights is equivalent to the MAXIMUM MATCHING problem; note that a matching of size k can be derived from a solution $V_1, \dots, V_k$ for 2-WEIGHT PACKING by choosing arbitrarily any edge in a set V_i with $|V_i| > 2$. In a similar way, the particular case of $W = 3$ is a problem studied under the names P_2-PACKING or PACKING 3-VERTEX PATHS (see e.g [36] and [23]). A $5k$ kernel for this problem is given in [26].

2 Balanced Expansion

In this section we introduce a balanced generalization of weighted expansions that we call *balanced expansion*.

Balanced expansion extends the existing weighted expansion structures and is one of the ingredients to derive our main BCD structure in the next section. Like the weighted expansion, it is a structure on bipartite graphs. We write $G = (A \cup B, E, w)$ for bipartite vertex-weighted graphs, where $w\colon A \cup B \to \mathbb{N}$ is its weight function. See Figure 1 for an illustration of this structure.

▶ **Definition 1** (*balanced expansion*). *Let $G = (A \cup B, E, w)$ be a bipartite vertex-weighted graph, where $w^B_{\max} = \max_{b \in B} w(b)$. For $q \in \mathbb{N}_0$, a partition $A_1 \cup A_2$ of A and $f\colon B \to A$, the tuple (A_1, A_2, f, q) is called a balanced expansion if:*

1. $w(a) + w\left(f^{-1}(a)\right) \begin{cases} \geq q - w^B_{\max} + 1, & a \in A_1 \\ \leq q + w^B_{\max} - 1, & a \in A_2 \end{cases}$
2. $f(b) \in N(b)$
3. $N\left(f^{-1}(A_1)\right) \subseteq A_1$

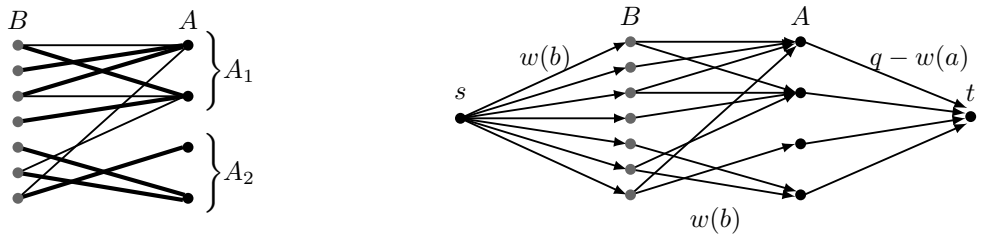

Figure 1 Left: Balanced expansion for $w(b) = 1$ for all $b \in B$, $q = 2$ and assignment f depicted with bold edges. Right: Flow network embedding of the graph on the left.

Our main result of this section is the following theorem.

▶ **Theorem 2** (*balanced expansion*). *Consider a vertex-weighted bipartite graph* $G = (A \cup B, E, w)$ *with no isolates in* B*, and* $q \geq \max_{b \in B} w(b) = w^B_{\max}$*. A balanced expansion* (A_1, A_2, f, q) *for* G *can be computed in* $\mathcal{O}(|V|\,|E|)$ *time. Furthermore, if* $w(A) + w(B) \geq q|A|$*, then* $A_1 \neq \varnothing$*.*

As intermediate structure we use a fractional version of the balanced expansion where we partially assign weights from vertices of B to vertices of A encoded as edge weights.

▶ **Definition 3** (*fractional balanced expansion*). *Let* $G = (A \cup B, E, w)$ *be a bipartite vertex-weighted graph. For* $q \in \mathbb{N}_0$*, a partition* $A_1 \cup A_2$ *of* A *and* $g\colon E \to \mathbb{N}_0$*, the tuple* (A_1, A_2, g, q) *is called fractional balanced expansion if:*

1. $w(a) + \sum_{b \in B} g(ab) \begin{cases} \geq q, & a \in A_1 \\ \leq q, & a \in A_2 \end{cases}$
2. $\forall b \in B\colon \sum_{a \in A} g(ab) \leq w(b)$ *(capacity)*
3. $N(B_U \cup B_{A_1}) \subseteq A_1$ *(separator)*

   where $B_a := \{b \in B \mid g(ab) > 0\}$ *for* $a \in A$*,* $B_{A'} := \bigcup_{a \in A'} B_a$ *for* $A' \subseteq A$ *and* $B_U := \left\{b \in B \mid \sum_{a \in A} g(ab) < w(b)\right\}$

We prove a fractional version of our result first in the following lemma.

▶ **Lemma 4** (*fractional balanced expansion*). *Given a vertex-weighted bipartite graph* $G = (A \cup B, E, w)$ *with no isolated vertices in* B *and* $q \in \mathbb{N}_0$*, a fractional balanced expansion* (A_1, A_2, g, q) *can be computed in* $\mathcal{O}(|V|\,|E|)$*. Also, if* $w(A) + w(B) \geq q|A|$*, then* $A_1 \neq \varnothing$*.*

Proof Sketch. The main idea is to embed G to a capacitated flow network in a standard way (see Figure 1). We construct a network $H = (A \cup B \cup \{s, t\}, \overrightarrow{E}, c)$. To obtain H from G, add source s and sink t, and arcs $\overrightarrow{E}$ with a capacity function $c\colon \overrightarrow{E} \to \mathbb{N}$ defined as follows. For every $b \in B$, add an arc $\overrightarrow{sb}$ with capacity $w(b)$ and for every $a \in A$, add an arc $\overrightarrow{at}$ with capacity $q - w(a)$. Moreover, transform every edge $ab \in E$ to an arc $\overrightarrow{ba}$ with capacity $w(b)$.

We compute a max flow $f\colon \overrightarrow{E} \to \mathbb{N}$ and define the saturated vertices $A' \subseteq A$ as $a \in A$ with $f(\overrightarrow{at}) = c(\overrightarrow{at})$. We now gradually build the sets A_1 and A_2. The vertices of A' are potential vertices for A_1 while the unsaturated vertices are immediately added to A_2. We define $F := \sum_{\overrightarrow{e} \in \delta^-(t)} f(\overrightarrow{e})$ as the flow value, where $\delta^-(v)$ denotes the incoming, and $\delta^+(v)$ the outgoing arcs for $v \in V(G)$. The final selection of A_1 follows by individually increasing the capacity by one for each $\overrightarrow{at}$ for $a \in A'$, and checking whether the flow value increases by computing a new max flow f_a with the increased capacity of $\overrightarrow{at}$. Let F_a be the flow value when the capacity of $\overrightarrow{at}$ is increased by one. If $F_a > F$, then the vertex is added to A_1, otherwise it is added to A_2. The intuition behind this selection can be explained as

follows: first observe that each $b \in B$ that has an edge ba_2 to some $a_2 \in A_2$ is saturated, i.e. $\sum_{\overrightarrow{e} \in \delta^+(b)} f(\overrightarrow{e}) = w(b)$. Otherwise, we could route an additional unit of flow from b to a_2 either in f or in f_{a_2}, giving a contradiction to the fact that $a_2 \in A_2$. Consequently, every unsaturated $b \in B$ is adjacent only to A_1. The second observation is that there are no $b \in B$ with $f(\overrightarrow{ba_1}) > 0$ and $ba_2 \in \overrightarrow{E}$ for $a_1 \in A_1$ and $a_2 \in A_2$. If such a b exist, we show that we can route an extra unit of flow from b to a_2 either in f or f_{a_2}. The idea is that we could reroute one unit of flow from $\overrightarrow{ba_1}$ to $\overrightarrow{ba_2}$ creating a vacuum for one unit of incoming flow in a_1. Since f_{a_1} routed one unit flow more than f, we could use a similar flow routing as in f_{a_1} to fill this vacuum, thus contradicting the maximality of either f or f_{a_2}. As a result, all vertices added to A_1 have the desired exclusive neighborhood in B encoded by f. Finally, in order to derive g we convert the flow arc values of f to edge weights for g. Note that the required upper bound on the assignment of A_2 follows from the capacities of the arcs from A to t, and the required lower bound on the assignment of A_1 follows from the vertices in A_1 being saturated. Regarding running time, we remark that it is sufficient to find one max-flow f at the beginning and then computing each f_a with only one augmenting flow step. The max-flow f can be computed in $\mathcal{O}(|V|\,|E|)$ time using the algorithm by Orlin [29]. ◀

Proof Sketch of Theorem 2. Once we have the fractional balanced expansion g, our first step is modifying the edge weights g such that the edge-weighted graph $G' := (V, \{ab \in E | g(a,b) > 0\}, g)$ becomes a forest, without changing the sum $\sum_{b \in N(a)} g(a,b)$ for any $a \in A$ and at the same time ensuring that $\sum_{a \in N(b)} g(a,b) \leq w(b)$ for all $b \in B$. This is possible through a standard cycle canceling process. Now consider the trees in this forest. The trees intersecting A_1 are disjoint from the trees intersecting A_2 due to the separation property of the balanced fractional expansion. For a tree T intersecting A_1, we allocate each $b \in V(T) \cap B$ completely to its parent in T. This way, any $a \in V(T) \cap A_1$ loses at most the assignment from its parent and hence its assignment decreases by at most $w^B_{\max} - 1$ as required. Now consider a tree T intersecting A_2. If a $b \in V(T) \cap B$ is a leaf of T its assignment has to be non-fractional, so it can be completely assigned to its parent a and deleted from the tree. This way, all leafs can be assumed to be from A_2. We then allocate each $b \in V(T) \cap B$ to one of its children, and thus to every $a \in V(T) \cap A_2$ at most the assignment from its parent is added, and hence the assignment increases by at most $w^B_{\max} - 1$ as required. ◀

Before moving to BCD, we formally state the aforementioned implication of the results in this section on the runtime of computing (non-balanced) expansions.

▶ **Lemma 5** (Weighted Expansion Lemma). *Let $G = (A \cup B, E)$ be a bipartite graph without isolated vertices in B, $w\colon B \to \{1, \ldots, W\}$, and $q \in \mathbb{N}_0$. A q-weighted expansion (f, H, C) in G can be computed in time $\mathcal{O}(|A \cup B|\,|E|)$. Furthermore, if $w(B) \geq q|A|$ then $H \neq \emptyset$.*

3 Balanced Crown Decomposition

In this section we introduce our combination of balanced connected partition and crown decomposition that we call *balanced crown decomposition*, formally defined as follows (see also Figure 2 for an illustration).

▶ **Definition 6.** *A **λ-balanced crown decomposition** (λ-BCD) of a vertex-weighted 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 $f\colon \mathbb{CC}(C) \to H$ where $\mathbb{CC}(C)$ is the set of connected components of $G[C]$, such that:*

1. *there are no edges from* C *to* R
2. $w(Q) < \lambda$ *for each* $Q \in \mathbb{CC}(C)$
3. $f(Q) \in N(Q)$ *for each* $Q \in \mathbb{CC}(C)$
4. $w(h) + w(f^{-1}(h)) \geq \lambda$ *for each* $h \in H$

(weighted crown dec.)

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

Our novel contribution is condition 5, that gives a balanced connected partition of the *body*. Without this condition, the structure is same as the weighted crown decomposition [40]. Observe that if there is a connected component of weight less than λ in G, then there is no λ-balanced crown decomposition for G. In the applications of BCD, such small components in the input are usually removed through some form of preprocessing.

We point out that the ratio 3 between the upper and lower bound in condition 5 of BCD is not arbitrary, but the best possible, since we want to ensure the existence of this structure in case all connected components have weight at least λ. A simple tight example is a triangle with each vertex having a weight of $\lambda - 1$; here, $C = H = \varnothing$ is the only possibility and hence $\mathfrak{R} = \{V\}$ is the only possible partition of $R = V$.

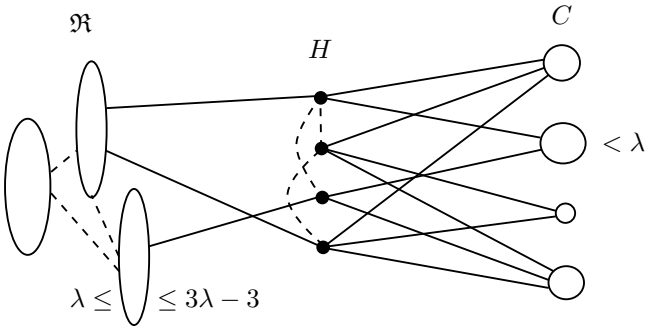

Figure 2 λ-balanced crown decomposition.

The main structural result of the paper is the following.

▶ **Theorem 7** (Balanced Crown Decomposition Theorem)**.** *Let* $G = (V, E, w)$ *be a vertex-weighted graph and* $\lambda \in \mathbb{N}$*, such that each connected component in* G *has weight at least* λ*. A* λ*-balanced crown decomposition* $(C, H, \mathfrak{R}, f)$ *of* G *can be computed in* $\mathcal{O}\left(k^2 \left|V\right| \left|E\right|\right)$ *time, where* $k = |H| + |\mathfrak{R}| \leq \min\{w(G)/\lambda, |V|\}$.

The proof of this result is very technical and we therefore here only give a very high-level overview of the ideas. Observe that the condition $|H| + |\mathfrak{R}| \leq \min\{w(G)/\lambda, |V|\}$ holds, since $\left\{\{h\} \cup f^{-1}(h) : h \in H\right\} \cup \mathfrak{R}$ is a partition of the vertices with each part having weight at least λ. This bound is also used to track our progress in our BCD algorithm. We maintain a set H that can be thought of as a *potential head* (not necessarily a separator), a set of connected components of weight smaller than λ (some of them assigned to vertices in H by a partial assignment f) which can be thought of as a *potential crown*, and a remaining *body* that is packed according to condition 5.

To easily talk about condition 5 in the following, we say $\mathcal{U}$ is a *connected packing* in $V' \subseteq V$, if for every $U \in \mathcal{U}$ we have $U \subseteq V'$, U induces a connected subgraph in G and $\bigcap_{U \in \mathcal{U}} U = \varnothing$. We say $\mathcal{U}$ is an $[a, b]$-*connected packing* of V' if $w(U) \in [a, b]$ for every $U \in \mathcal{U}$ and that $\mathcal{U}$ is *maximal* if the remaining graph does not have a connected component of weight at least a. Recall that we say $\mathcal{U}$ is a CVP or $[a, b]$-CVP of V' if additionally $\bigcup_{U \in \mathcal{U}} U = V'$ holds, and observe that condition 5 asks for a $[\lambda, 3\lambda - 3]$-CVP of the body R.

Proof Sketch of Theorem 7. Let $G = (V, E, w)$ be a vertex-weighted graph and $\lambda \in \mathbb{N}$ such that each connected component of G has weight at least λ. We reduce G by deleting all vertices of weight more than λ and all connected components of size smaller than λ that occur after this deletion. Suppose we have a λ-BCD for the reduced graph, then a λ-BCD of G can be built by adding the deleted heavy vertices to the head, the deleted small components to the crown and assigning (by the function f in the definition of λ-BCD) each of these components arbitrarily to a heavy vertex it is adjacent to. Thus, we can assume that every vertex has weight at most λ. See Figure 3 for an illustration of the structures arising below.

We start with a maximal $[\lambda, 2\lambda]$-connected packing of G which is obtained greedily (slight deviation from the main proof to provide better intuition). Let $\mathfrak{R} = \{R_1, R_2, \cdots\}$ be this packing, C be the vertices not in the packing, and let $\mathbb{CC}(C) = \{C_1, C_2, \cdots\}$ be the connected components of $G[C]$. Note that by the maximality of the packing, $w(C_i) < \lambda$ for all i. Think of $\mathfrak{R}$ as the current body and C as the current crown, and the head is empty in the beginning. Note that at this point we do not ensure that there are no edges between crown and body. If C is empty then we already have a λ-BCD (with empty crown and head). Also, if we can somehow assign each C_i to some adjacent R_i such that each R_i is assigned weight at most 3λ (including its own weight), then we have also built a λ-BCD (with empty crown and head). Assuming neither of these cases hold, there has to exist an R_i such that its weight plus the weight of the neighborhood in the crown part is at least 3λ; recall that we assumed that all connected components of G have weight at least λ, so each C_j is connected to at least one component in $\mathfrak{R}$. We call the subgraph induced by R_i together with all C_j that are connected to it the *effective neighborhood of* R_i, and its weight the *effective weight of* R_i.

In case we have not found a λ-BCD yet, we pick an R_i with effective weight at least 3λ and use the following fact derived from the famous results of Tarjan [33, 16]: for any connected graph of total weight at least 3λ and largest vertex weight at most λ, we can efficiently either find a partition of it into two connected subgraphs of weight at least λ each, or find a cut-vertex that cuts the graph into components each of weight less than λ. If the effective neighborhood is divided into two, we take each of the two parts into the body and remove R_i, thus increasing the body size by one. In the other case, that is, if we find a cut-vertex c, then we add c to the head and the components of $R_i \setminus \{c\}$ (each having weight less than λ) to the crown $\mathbb{CC}(C)$. We assign with a *partial* function $f\colon \mathbb{CC}(C) \to H$ some of these components that we just added to $\mathbb{CC}(C)$ to c such that c is assigned a total weight of at least 3λ (including weight of c). The reason for assigning 3λ when we only require λ by the definition of BCD, will become clear in the following. The new components added to the crown could have edges to the old components there and hence can merge with these. If at any point it happens that there is a component of weight at least λ in the crown, then we immediately add it to the body. This could cause some head vertex to loose some of its assignment, but since it had 3λ assigned to it, an assignment of at least λ remains. This is because we ensure that the part we move to the body can have weight at most 2λ, as we move it immediately as the weight is at least λ, and each addition is by steps of less than λ.

We repeat this process of picking an effective neighborhood of an R_i and dividing or cutting it. We point out that when we calculate effective neighborhoods and weights, we do not consider the crown parts that are already assigned by f. This process continues until the effective weights of all sets R_i are less than 3λ. We claim that the reason why we have not arrived at a λ-BCD yet could only be that there are crown parts that do not have edges to the body (we call them *private components*) and *not* assigned by f, while there are also crown parts that have edges to the body (non-private components) and assigned by f. Note that if there are no unassigned private components, we can merge all unassigned crown

parts with some set in the body and create a $[\lambda, 3\lambda]$-CVP given by the R_i's and the sets $\{v\} \cup f^{-1}(v)$. Note that since the effective weights were lighter than 3λ, the body parts after the merging are lighter than 3λ. Also, if f does not assign any non-private components, we can assign unassigned private components to arbitrary neighbors in H, and merge unassigned non-private components to body obtaining a λ-BCD.

We modify the assignment f to switch non-private with private components in the best way possible. For this we use the balanced expansion Theorem 2. We build the bipartite graph where the set A are the head vertices, and the set B are the private crown components each contracted into a single vertex. Theorem 2 with expansion parameter 2λ then gives us sets $A' \subseteq A$ and $B' \subseteq B$ and an assignment f' such that $w(\{a'\} \cup f'^{-1}(a')) \geq \lambda$ for all $a' \in A'$ and $w(\{a\} \cup f'^{-1}(a)) \leq 3\lambda$ for all $a \in A \setminus A'$, and the crown components in B' are completely assigned to A' and do not have neighbors in $A \setminus A'$. Note that since B was the set of private components, the components B' do not have neighbors in the body either. Now augment f' by assigning to $A \setminus A'$ also enough non-private components such that they have an assignment of at least 3λ each. This is possible since each vertex in $A \setminus A'$ has an assignment of 3λ by f which did not use any components from B' (as there are no edges from B' to $A \setminus A'$). Note that this augmentation of f' needs to be done carefully since the private components could be assigned by the balanced expansion differently than by f.

By f' all private components are now assigned to the head, but there could still be non-private ones assigned as well. But now, if the effective weight of each R_i is at most 3λ, we can add the unassigned crown parts to sets in $\mathfrak{R}$, and thus create a λ-BCD: A' with its assignment by f' are head and crown, and $\mathfrak{R}$ plus the sets $\{a\} \cup f'^{-1}(a)$ with $a \in A \setminus A'$ are a $[\lambda, 3\lambda]$-CVP of the body. Thus, if we are not successful, there exists an R_i with effective weight more than 3λ and we continue by dividing or cutting it. Note that we can proceed with f' replacing f although some head vertices (from what was A' in the balanced expansion) might only have weight λ assigned to them (and not 3λ), because the crown parts assigned to them are private and hence do not interfere with the further process.

To analyse the run time, we estimate how often we divide or cut a set R_i; note that each such step can be performed in $\mathcal{O}(|V||E|)$. Throughout our algorithm, the value $|H| + |\mathfrak{R}|$ is non-decreasing, and upper bounded by k. Every time we divide some R_i, we increase $|H| + |\mathfrak{R}|$, hence this happens at most k times. Every time we cut some R_i we increase $|H|$ by one. Since $|H|$ is also upper bounded by k, and we are careful not to decrease $|H|$ with the balancing step in-between cut steps, we arrive at a total of at most k^2 divide or cut steps.

One pitfall here is that after applying the balanced expansion one might be tempted to just take A' and its assignment via f' into head and crown respectively, delete it, and start over on the rest of the graph. The problem with this is that we are not guaranteed to find a non-empty set A' (since the private components might not have weight at least $\lambda|A|$). The way we augment f', we ensure that we retain the preliminary crown, head and body structure, and with this especially the value $|H|$, and can split up another R_i to either increase $|H|$ or $|H| + |\mathfrak{R}|$. Further, the reason why *we cannot use the standard weighted expansion lemma* here is that we would need a lower bound of at least $\lambda|A|$ on the weight of B for this. We cannot ensure that the private components of the crown alone have a weight of at least $\lambda|A|$, since we also used the non-private components for the assignment f.

One detail that we did not mention so far is that it is not possible to assign exactly 3λ to each head vertex. Since the step size we can guarantee is only λ, we might have to assign $(4\lambda - 1)$ in order to get a value of at least 3λ. Recall that we assign a collection of components of weight less than λ. Without further work, this only yields an upper bound of 4λ instead of 3λ for the packing of the body, worsening the quality of our structure (we for

example would only get a 4- instead of a 3-approximation for the BCP problems). For this improvement from 4 to 3, we maintain a "last component" as a special assignment. (This is called g'-assignment in the full proof). The details of how we make use of this special second assignment is rather technical, and to some extent complicates the proof. If one is satisfied with a bound of 4λ for the body, this complication is not necessary.

Another technical detail we skipped is that in the assignment f we maintain, the crown parts we map may not be whole sets in $\mathbb{CC}(C)$ (connected components induced by crown vertices). They are connected, but can be subgraphs of some $C_i \in \mathbb{CC}(C)$. We call such subgraphs *sub-components*. Different sub-components of some C_i can be assigned to different heads. Also, for a C_i some of its sub-components can be assigned while others are not. For our structure to converge to a λ-BCD, sub-components have to be classified as private or non-private based on the set $\mathbb{CC}(C)$ they are a part of, so it can happen that a sub-component is non-private but has no edge to the body. Whenever we make the move from crown to body, we therefore have to do a merging of some sub-components such that for each $C_i \in \mathbb{CC}(C)$ either all its sub-components are assigned to the head or none of them are. ◀

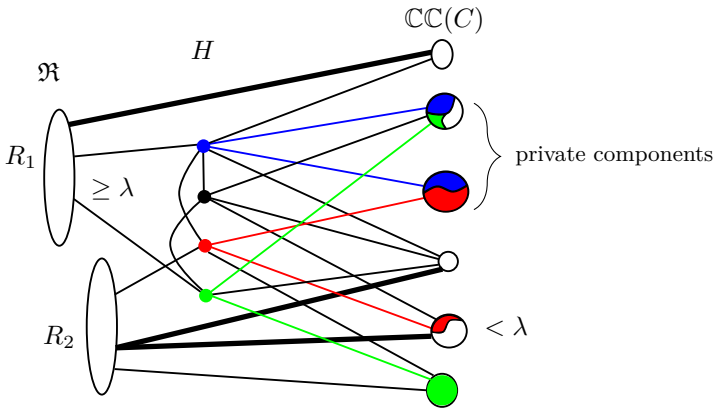

Figure 3 Illustration of a possible intermediate stage in the proof of Theorem 7. Colors represent the partial assignment f, e.g., the two blue-colored sub-components are assigned to the blue vertex in H. Thick lines are edges that go from the sets of the body to their effective neighborhoods.

4 Applications of Balanced Crown Decomposition

In this section we present some applications of the balanced crown decomposition.

For the problems W-WEIGHT SEPARATOR and W-WEIGHT PACKING we immediately get the following theorems by reducing an instance (G, k, W) by first finding a W-BCD $(C, H, \mathfrak{R}, f)$ of G, and then applying the standard crown reduction rule that removes the head H and crown C from G. We emphasize that the balanced connected partition of the body is crucial to obtain the kernel sizes. These are the first kernels for vertex-weighted graphs, while also improving the state-of-the-art results for the unweighted cases.

▶ **Theorem 8.** *W-WEIGHT SEPARATOR admits a kernel of weight $3k\,(W-1)$. Furthermore, such a kernel can be computed in time $\mathcal{O}\left(k^2|V|\,|E|\right)$.*

▶ **Theorem 9.** *W-WEIGHT PACKING admits a kernel of weight $3k\,(W-1)$. Furthermore, such a kernel can be computed in time $\mathcal{O}\left(k^2|V|\,|E|\right)$.*

For the optimization variant of the W-WEIGHT PACKING problem, i.e. maximizing the size of packing, the fact that the partition $\mathfrak{R}$ is a solution also gives a 3-approximation; to the best of our knowledge, the first approximation result for the problem.

▶ **Theorem 10.** *A 3-approximation for the optimization problem of W-WEIGHT PACKING can be computed in $\mathcal{O}\left(k^{*2}|V|\,|E|\right)$, where k^* denotes the optimum value.*

To better sketch the ideas for our results for the BCP problems, we denote by I-CVP$_k$ for an interval I, a CVP with k parts where each part has a weight in I. We derive the following result for MAX-MIN BCP, which is the first constant approximation for this problem.

▶ **Theorem 11.** *A 3-approximation for the MAX-MIN BCP problem can be computed in $\mathcal{O}\left(\log(X^*)\,k^2|V|\,|E|\right)$, where X^* denotes the optimal value.*

Proof. Let (G,k) be an instance of MAX-MIN BCP. For any value X, using BCD, we show how to either obtain an $[X/3,\infty)$-CVP$_k$, or report that $X > X^*$. Once we have this procedure in hand, a binary search can be used to obtain an $[X^*/3,\infty)$-CVP$_k$.

We first obtain a λ-BCD $(C,H,\mathfrak{R},f)$ of G with $\lambda = \lceil X/3 \rceil$. If $|H| + |\mathfrak{R}| \geq k$, we output a $[X/3,\infty)$-CVP$_k$ given by the body and the assignment to head vertices (if this gives more than k sets, arbitrarily merge some until there are only k). If $|H| + |\mathfrak{R}| < k$, then we report that $X > X^*$. To see that this is correct, assume towards contradiction that $X \leq X^*$, and consider an optimal solution $\mathcal{S}^* = \{S_1^*,\ldots,S_k^*\}$. Then in the λ-BCD we computed, we know that $w(R) < X$ for every $R \in \mathfrak{R}$ and $w(C') < X$ for every $C' \in \mathbb{CC}(C)$. Observe that then no $C' \in \mathbb{CC}(C)$ or a subset of it can be a set in $\mathcal{S}^*$, since $w(S_i^*) \geq X^* \geq X$ for every $S_i^* \in \mathcal{S}$. From the separator properties of H and that the fact that each $S_i^* \in \mathcal{S}$ is connected, we obtain that any set in $\mathcal{S}^*$ containing vertices from C also has to contain at least one vertex from H. Thus, we can derive that the cardinality of $\mathcal{S}_H^* = \{S_i^* \in \mathcal{S}^* | S_i^* \cap (C \cup H) \neq \varnothing\}$ is at most $|H|$. Also, $|\mathcal{S}^* \setminus \mathcal{S}_H^*| \leq w(V(\mathfrak{R}))/X^* \leq w(V(\mathfrak{R}))/X \leq |\mathfrak{R}|$. Thus it follows that $|\mathcal{S}^*| \leq |H| + |\mathfrak{R}| < k$, a contradiction. ◀

The last problem that we consider as application of the balanced crown decomposition is the MIN-MAX BCP problem, where we also provide the first constant approximation result.

▶ **Theorem 12.** *A 3-approximation for the MIN-MAX BCP problem can be computed in time $\mathcal{O}\left(\log(X^*)\,|V|\,|E|\left(\log\log X^* \log(|V|w_{max}) + k^2\right)\right)$, where X^* denotes the optimum value and $w_{max} = \max_{v\in V} w(v)$ the maximum weight of a vertex.*

Proof. Achieving this requires several technical steps after having a balanced crown decomposition in hand, including a second use of the balanced expansion. Let (G,k) be an instance of MIN-MAX BCP and let $\mathcal{S}^* = \{S_1^*,\ldots,S_k^*\}$ be an optimal solution. Let $(C,H,\mathfrak{R},f)$ be a λ-BCD of G. Similar to the Max-Min case we try to make a comparison between $\mathcal{S}^*$ and the vertex decomposition C,H and $V(\mathfrak{R})$. The main issue is that, in contrast to the Max-Min case, an optimal solution can (and sometimes has to) build more than $|H|$ sets from the vertices in $H \cup C$. With the connectivity constraints, this means that some components in $G[C]$ are in fact a set in the optimal partition. Hence, when computing an approximate solution from a balanced crown decomposition, we have to also choose some components from $G[C]$ to be sets, while others are combined with some vertex in H. In order to make the decision of where to place the components in $G[C]$, we compute a min-cost flow using the algorithm from [1] on a network that models the options for components in $G[C]$ to either be sets or be combined with some vertex in H. A partial embedding of $\{S_i^* \in \mathcal{S}^* | S_i^* \cap (C \cup H) \neq \varnothing\}$ to this cost-flow network allows a comparison with the resulting

partition of $C \cup H$. The balanced weight properties of $\mathfrak{R}$ then yield a comparison with the whole set S^*. With the additional use of a min cost-flow network, our balanced crown structure can be used to estimate the optimal objective value and again enables a binary search for an approximate solution. ◀

References

1 Ravindra K Ahuja, Andrew V Goldberg, James B Orlin, and Robert E Tarjan. Finding minimum-cost flows by double scaling. *Mathematical programming*, 53(1-3):243–266, 1992.

2 Curtis A Barefoot, Roger Entringer, and Henda Swart. Vulnerability in graphs a comparative survey. *Journal of Combinatorial Mathematics and Combinatorial Computing*, 1:13–22, 1998.

3 Ralf Borndörfer, Ziena Elijazyfer, and Stephan Schwartz. Approximating balanced graph partitions. Technical Report 19-25, ZIB, Takustr. 7, 14195 Berlin, 2019.

4 Christoph Brause and Ingo Schiermeyer. Kernelization of the 3-path vertex cover problem. *Discrete Mathematics*, 339(7):1935–1939, 2016. doi:10.1016/j.disc.2015.12.006.

5 Paolo M Camerini, Giulia Galbiati, and Francesco Maffioli. On the complexity of finding multi-constrained spanning trees. *Discrete Applied Mathematics*, 5(1):39–50, 1983.

6 L. Sunil Chandran, Yun Kuen Cheung, and Davis Issac. Spanning tree congestion and computation of generalized győri-lovász partition. In *45th International Colloquium on Automata, Languages, and Programming*, volume 107 of *LIPIcs*, pages 32:1–32:14, 2018.

7 Frédéric Chataigner, Liliane Benning Salgado, and Yoshiko Wakabayashi. Approximation and inapproximability results on balanced connected partitions of graphs. *Discrete Mathematics and Theoretical Computer Science*, 9(1), 2007. URL: http://dmtcs.episciences.org/384.

8 Guangting Chen, Yong Chen, Zhi-Zhong Chen, Guohui Lin, Tian Liu, and An Zhang. Approximation algorithms for the maximally balanced connected graph tripartition problem. *Journal of Combinatorial Optimization*, pages 1–21, 2020.

9 Jianer Chen, Henning Fernau, Peter Shaw, Jianxin Wang, and Zhibiao Yang. Kernels for packing and covering problems. In *Frontiers in Algorithmics and Algorithmic Aspects in Information and Management*, pages 199–211. Springer, 2012.

10 Jiangzhuo Chen, Robert D Kleinberg, László Lovász, Rajmohan Rajaraman, Ravi Sundaram, and Adrian Vetta. (almost) tight bounds and existence theorems for single-commodity confluent flows. *Journal of the ACM (JACM)*, 54(4):16, 2007.

11 Yong Chen, Zhi-Zhong Chen, Guohui Lin, Yao Xu, and An Zhang. Approximation algorithms for maximally balanced connected graph partition. In *International Conference on Combinatorial Optimization and Applications*, pages 130–141. Springer, 2019.

12 Janka Chlebíková. Approximating the maximally balanced connected partition problem in graphs. *Information Processing Letters*, 60(5):225–230, 1996.

13 Benny Chor, Mike Fellows, and David W. Juedes. Linear kernels in linear time, or how to save k colors in o(n^2) steps. In *Graph-Theoretic Concepts in Computer Science, 30th International Workshop*, volume 3353 of *LNCS*, pages 257–269. Springer, 2004.

14 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.

15 Rosa Enciso, Michael R Fellows, Jiong Guo, Iyad Kanj, Frances Rosamond, and Ondřej Suchỳ. What makes equitable connected partition easy. In *International Workshop on Parameterized and Exact Computation*, pages 122–133. Springer, 2009.

16 Shimon Even and Robert Endre Tarjan. Computing an st-numbering. *Theoretical Computer Science*, 2(3):339–344, 1976.

17 Fedor V. Fomin, Daniel Lokshtanov, Neeldhara Misra, Geevarghese Philip, and Saket Saurabh. Hitting forbidden minors: Approximation and kernelization. *SIAM Journal on Discrete Mathematics*, 30(1):383–410, 2016.

18 Fedor V Fomin, Daniel Lokshtanov, Saket Saurabh, and Meirav Zehavi. *Kernelization: theory of parameterized preprocessing*. Cambridge University Press, 2019.

19 Greg N. Frederickson. Optimal algorithms for tree partitioning. In *Proceedings of the Second Annual ACM/SIGACT-SIAM Symposium on Discrete Algorithms*, pages 168–177. ACM/SIAM, 1991. URL: http://dl.acm.org/citation.cfm?id=127787.127822.

20 E Gyori. On division of graphs to connected subgraphs, combinatorics. In *Colloq. Math. Soc. Janos Bolyai, 1976*, 1976.

21 Alexander Hoyer. *On the Independent Spanning Tree Conjectures and Related Problems*. PhD thesis, Georgia Institute of Technology, 2019.

22 Jörg Kalcsics and Roger Z. Ríos-Mercado. *Districting Problems*, pages 705–743. Springer International Publishing, Cham, 2019.

23 Atsushi Kaneko, Alexander K. Kelmans, and Tsuyoshi Nishimura. On packing 3-vertex paths in a graph. *Journal of Graph Theory*, 36(4):175–197, 2001. doi:10.1002/1097-0118(200104)36:4\%3C175::AID-JGT1005\%3E3.0.CO;2-T.

24 Mithilesh Kumar and Daniel Lokshtanov. A 2lk kernel for l-component order connectivity. In *11th International Symposium on Parameterized and Exact Computation*, volume 63 of *LIPIcs*, pages 20:1–20:14, 2016. doi:10.4230/LIPIcs.IPEC.2016.20.

25 Sukhamay Kundu and Jayadev Misra. A linear tree partitioning algorithm. *SIAM Journal on Computing*, 6(1):151–154, 1977.

26 Wenjun Li, Junjie Ye, and Yixin Cao. Kernelization for p_2-packing: A gerrymandering approach. In *Frontiers in Algorithmics - 12th International Workshop*, volume 10823 of *LNCS*, pages 140–153. Springer, 2018. doi:10.1007/978-3-319-78455-7_11.

27 Richard J. Lipton and Robert Endre Tarjan. A separator theorem for planar graphs. *SIAM Journal of Applied Mathematics*, 36:177—-189, 1979.

28 László Lovász. A homology theory for spanning tress of a graph. *Acta Mathematica Academiae Scientiarum Hungarica*, 30(3-4):241–251, 1977.

29 James B Orlin. Max flows in $o(nm)$ time, or better. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 765–774, 2013.

30 Yehoshua Perl and Stephen R Schach. Max-min tree partitioning. *Journal of the ACM (JACM)*, 28(1):5–15, 1981.

31 Arnold L. Rosenberg and Lenwood S. Heath. *Graph separators with applications*. Frontiers of computer science. Springer, 2001.

32 Hitoshi Suzuki, Naomi Takahashi, and Takao Nishizeki. A linear algorithm for bipartition of biconnected graphs. *Information Processing Letters*, 33(5):227–231, 1990.

33 Robert Endre Tarjan. Depth-first search and linear graph algorithms. *SIAM Journal on Computing*, 1(2):146–160, 1972.

34 Jianhua Tu, Lidong Wu, Jing Yuan, and Lei Cui. On the vertex cover p_3 problem parameterized by treewidth. *Journal of Combinatorial Optimization*, 34(2):414–425, 2017. doi:10.1007/s10878-016-9999-6.

35 Koichi Wada and Kimio Kawaguchi. Efficient algorithms for tripartitioning triconnected graphs and 3-edge-connected graphs. In *International Workshop on Graph-Theoretic Concepts in Computer Science*, pages 132–143. Springer, 1993.

36 Jianxin Wang, Dan Ning, Qilong Feng, and Jianer Chen. An improved kernelization for p_2-packing. *Information Processing Letters*, 110(5):188–192, 2010. doi:10.1016/j.ipl.2009.12.002.

37 Lele Wang, Zhao Zhang, Di Wu, Weili Wu, and Lidan Fan. Max-min weight balanced connected partition. *Journal of Global Optimization*, 57(4):1263–1275, 2013.

38 Bang Ye Wu. A 7/6-approximation algorithm for the max-min connected bipartition problem on grid graphs. In *International Conference on Computational Geometry, Graphs and Applications*, pages 188–194. Springer, 2010.

39 Bang Ye Wu. Fully polynomial-time approximation schemes for the max–min connected partition problem on interval graphs. *Discrete Mathematics, Algorithms and Applications*, 4(1), 2012.

40 Mingyu Xiao. Linear kernels for separating a graph into components of bounded size. *Journal of Computer and System Sciences*, 88:260–270, 2017.

41 Mingyu Xiao and Shaowei Kou. Kernelization and parameterized algorithms for 3-path vertex cover. In *Theory and Applications of Models of Computation*, volume 10185 of *LNCS*, pages 654–668, 2017. doi:10.1007/978-3-319-55911-7_47.

42 Mihalis Yannakakis. Node-deletion problems on bipartite graphs. *SIAM Journal on Computing*, 10(2):310–327, 1981. doi:10.1137/0210022.

43 Xing Zhou, Huaimin Wang, Bo Ding, Tianjiang Hu, and Suning Shang. Balanced connected task allocations for multi-robot systems: An exact flow-based integer program and an approximate tree-based genetic algorithm. *Expert Systems with Applications*, 116:10–20, 2019.