The List Allocation Problem and Some of its Applications in Parameterized Algorithms

Eunjung Kim[2], Sang-il Oum[1], Christophe Paul[3], Ignasi Sau[3], and Dimitrios M. Thilikos[3,4]

[1] Department of Mathematical Sciences, KAIST, Daejeon, South Korea.
sangil@kaist.edu
[2] CNRS, LAMSADE, Paris, France.
eunjungkim78@gmail.com
[3] AlGCo project team, CNRS, LIRMM, Montpellier, France.
christophe.paul@lirmm.fr, ignasi.sau@lirmm.fr, sedthilk@thilikos.info
[4] Department Mathematics, University of Athens, Greece.

Abstract. In this article we introduce a parameterized cut problem in graphs, called List Allocation, and prove that it is Fixed-Parameter Tractable (FPT). Our algorithm uses, in particular, a sequence of Turing FPT-reductions and several ingredients of the *randomized contraction* technique introduced by Chitnis *et al.* [FOCS 2012]. Besides being a natural and quite general cut problem by itself and encompassing, in particular, the Multiway Cut problem, the relevance of List Allocation is best demonstrated by the following algorithms, which we obtain by reducing in FPT time each corresponding problem to particular cases, or slight variations, of List Allocation:

- An FPT-algorithm for the Min-Max-Multiway Cut problem, which is the variation of Multiway Cut where the parameter is the maximum number of edges (instead of the sum) leaving any connected component defined by the edge cut.
- FPT-algorithms for the Edge Cutting into Many Components and Cutting a Specific Number of Vertices problems, which are the natural vertex variants of two cut problems introduced by Marx [TCS 2006] and that were known to be W[1]-hard for some other parameterizations.
- An FPT-algorithm for a generalization of Digraph Homomorphism, which we call Arc-Specified List Digraph Homomorphism, where given two digraphs G and H, a list of allowed vertices of H for every vertex of G, and a prescribed number of arcs of G to be mapped to every non-loop arc of H, one has to decide whether there exists a homomorphism from G to H respecting these constraints.
- An FPT-algorithm for computing a 2-approximation of the *tree-cut width* of a graph, a graph invariant recently introduced by Wollan [JCTB 2015] and that has proved of fundamental importance in the structure of graphs not admitting a fixed graph as an immersion. Obtaining this FPT 2-approximation requires some additional algorithmic steps, and is one of the main technical contributions of this article.

Keywords: Parameterized complexity; graph cut; Fixed-Parameter Tractable algorithm; digraph homomorphism; graph immersion; tree-cut width.

1 Introduction

The Multiway Cut problem asks, given a graph G, a set of r terminals T, and a non-negative integer w, whether it is possible to partition $V(G)$ into r parts such that each part contains exactly one of the terminals of T and there are at most w edges among different parts (i.e., at most w crossing edges). In the special case where $|T| = 2$, this gives the celebrated Minimum Cut problem, which is polynomially solvable [28]. In general, when there is no restriction on the number of terminals, the Multiway Cut problem is NP-complete [9] and a lot of research has been devoted to the study of this problem and its generalizations, including several classic results on its polynomial approximability More recently, more attention to the Multiway Cut problem was given from the parameterized complexity point of view. The existence of an FPT-algorithm, i.e., an $f(w) \cdot n^{O(1)}$-step algorithm, for Multiway Cut had been an long-standing open problem. This question was answered positively by Marx in [23] with the use of the *important separators technique*. The same technique was later useful for the design of FPT-algorithms for several other variants and generalizations of the Multiway Cut problem [4–6].

1.1 List Allocation

In this paper we introduce a vast generalization of the Multiway Cut problem, namely the List Allocation problem. List Allocation encompasses Multiway Cut and is general enough in the sense that several other, quite diverse problems, are TFPT-reducible[1] to it.

The List Allocation problem is defined as follows: We are given a graph G and a set of r "boxes" indexed by numbers from $\{1, \ldots, r\}$. Each vertex v of G is accompanied with a list $\lambda(v)$ of indices corresponding to the boxes where it is allowed to be allocated. Moreover, there is a weight function α assigning to every pair of different boxes a non-negative integer. The question is whether there is a way to place each of the vertices of G into some box of its list such that, for any two different boxes i and j, the number of crossing edges between them is *exactly* $\alpha(i, j)$. By a straightforward reduction from Maximum Cut, it follows that List Allocation is NP-complete, even when $r = 2$. Throughout this paper, we parameterize the List Allocation problem by the total number w of "crossing edges" between different boxes, i.e., $w = \sum_{1 \leq i < j \leq r} \alpha(i, j)$.

Let us verify first that Multiway Cut, parameterized by w, is TFPT-reducible to List Allocation. Given an instance of Multiway Cut, we first discard from its graph all the connected components that have at most 1 terminal. Clearly, this gives an equivalent instance $(G, T = \{t_1, \ldots, t_r\}, w)$ where $r \leq w+1$. Next, we consider the set $\mathcal{A}$ containing every weight function α where $\sum_{1 \leq i < j \leq r} \alpha(i, j) \leq w$. Let also $\lambda : V(G) \to 2^{[r]}$ be the list function such that if

[1] Let **A** and **B** be two parameterized problems. We say that a parameterized problem **A** is *Turing* FPT*-reducible* to **B** when the existence of an FPT-algorithm for **B** implies the existence of an FPT-algorithm for **A**. (For brevity, in this paper, we write "TFTP" instead of "Turing FPT".)

$v = t_i \in T$, then $\lambda(v) = \{i\}$, otherwise $\lambda(v) = \{1, \ldots, r\}$. It is easy to verify that (G, T, w) is a YES-instance of Multiway Cut if and only if there exists some $\alpha \in \mathcal{A}$ such that (G, r, λ, α) is a YES-instance of List Allocation. This yields the claimed reduction, as $|\mathcal{A}|$ is clearly bounded by some function of w. The above reduction to the List Allocation problem turns out to be quite versatile. As we will see in Subsection 1.3, by suitably adapting the definition of λ and the set $\mathcal{A}$, we can easily TFPT-reduce more problems to List Allocation.

Our main result is an FPT-algorithm for the List Allocation problem. In particular, we give an algorithm that solves this problem in $2^{O(w^2 \log w)} \cdot n^4 \cdot \log n$ steps.

1.2 Techniques

The proof that List Allocation admits an FPT-algorithm is the consequence of a series of TFPT-reductions between several variants of the problem. Briefly, these reductions are the following:

1. List Allocation is TFPT-reduced to its restriction, called CLA, where G is a connected graph and only $O(w)$ boxes are used. This reduction takes care about the different ways connected components of G can entirely by placed into the boxes (see Subsection 3.1).
2. CLA is TFPT-reduced to a restriction of it, called HCLA, where G is highly connected in the sense that there is no set of $w - 1$ edges that can separate G into two parts of at least $f(w)$ vertices each (for some suitable function f). This reduction is presented in Subsection 3.2 and uses the technique of *recursive understanding*, introduced in [6] and further developed in [8] (see also [19]), for generalizations of the Multiway Cut problem.
3. HCLA is TFPT-reduced to a special enhancement of it, called SHCLA, whose input additionally contains some set $S \subseteq V(G)$ and the problem asks for a solution where all vertices of S are placed in a unique "big" box and all vertices of this box that are incident to crossing edges between different boxes are contained in S. This variant of the problem permits the application of the technique of *randomized contraction*, also introduced in [6] (see Subsection 3.3).
4. Finally, SHCLA is TFPT-reduced to List Allocation restricted to instances whose sizes are bounded by a function of the parameter. This algorithmic reduction is presented in Subsection **??** and is based on the fact that an essentially equivalent instance of the problem can be constructed if, apart from S, we remove from G all but a bounded number of the connected components of $G \setminus S$.

The specification of the parametric dependencies in all the above reductions yields the claimed running time.

1.3 Consequences and applications

Our main result has several consequences on quite diverse parameterized problems. Due to space restrictions, their presentation has entirely been moved to Appendix B. We list them below:

Min-Max-Multiway Cut. In the Multiway Cut problem the parameter w bounds the total number of crossing edges (i.e., edges with endpoints in different parts). Svitkina and Tardos [29] considered a "min-max" variant of this problem, namely the Min-Max-Multiway Cut, where w bounds the maximum number of outgoing edges of the parts, i.e., crossing edges that have endpoints in the same part. (Notice that under this viewpoint Multiway Cut can be seen as Min-Sum-Multiway Cut.) In [29], it was proved that Min-Sum-Multiway Cut is NP-complete even for fixed $r = 4$. As a consequence of the results in [29] and [25], Min-Max-Multiway Cut admits an $O(\log^2 n)$-approximation algorithm. This was improved recently in [1] to a $O((\log n \cdot \log r)^{1/2})$-approximation algorithm. To our knowledge, nothing is known about the parameterized complexity of this problem.

Our first result is that Min-Sum-Multiway Cut admits an FPT-algorithm when parameterized by both r and w. The proof is almost the same TFPT-reduction to the List Allocation problem as the one we described before for Multiway Cut. The only difference is that we now define $\mathcal{A}$ so to contain every α such that $\forall i \in \{1, \ldots, w\},\ \sum_{j \in \{1,\ldots,w\} \setminus \{i\}} \alpha(i,j) \leq w$. As a result of this, Min-Sum-Multiway Cut can be solved in $2^{O((wr)^2 \log wr)} \cdot n^4 \cdot \log n$ steps. The details are presented in Subsection B.1.

Edge Cutting into Many Components. The input of the Edge Cutting into Many Components problem (ECMC for short) is a graph G and two non-negative integers w and r. The question asks whether there is a set of at most w edges in G whose removal leaves at least r connected components. The "vertex variant" of this problem was studied by Marx in [23], where it was proven to be W[1]-hard when parameterized by w or r, while it admits an FPT-algorithm when parameterized by both w and r. It appears that the landscape in the case of ECMC is somewhat different. According to [13], this problem is W[1]-hard when parameterized by r, and in this paper we prove that it admits an FPT-algorithm when parameterized by w. In order to expose the versatility of our approach, we examine the case where G is a connected graph. Indeed, this assumption permits us to assume that $r \leq w + 1$ (otherwise we have a NO-instance). Then the problem is TFPT-reducible to List Allocation by the same reduction scheme as in the case of Multiway Cut, with the difference that now we set $\lambda(v) = \{1, \ldots, r\}$ for all $v \in V(G)$ (i.e., no list restrictions are imposed) and $\mathcal{A}$ contains every α such that $\sum_{1 \leq i < j \leq r} \alpha(i,j) \leq w$ and $\forall i \in \{1, \ldots, r\},\ \exists j \in \{1, \ldots, r\} \setminus \{i\}$ such that $\alpha(i,j) > 0$. The general, non-connected, setting is treated in Subsection B.2, where we prove that ECMC can be solved in $2^{O(w^2 \cdot \log w)} \cdot n^4 \cdot \log n$ steps.

Cutting a Specific Number of Vertices. In the Cutting a Specific Number of Vertices problem (CSNV) we are given a graph and two non-negative integers r and w, and the question is whether G has a set S of *exactly* r vertices such that the edges with one endpoint in S and the other outside S are at most w. The variant of the above problem where we additionally demand that $G[S]$ is a connected graph is called Cutting a Specific Number of Connected Vertices (CSNCV).

In the vertex-separation analogue of Cutting a Specific Number of Vertices we are asked for a partition $\{S, B, A\}$ of $V(G)$ such that $|B| \leq w$, $|S| = r$, there is no edge with one endpoint in S and the other in S, and there are at most w "outgoing edges" from S. This problem is called Separating a Specific Number of Vertices (SSNV) and, its connected analogue, where we demand $G[S]$ to be connected, is called Separating a Specific Number of Connected Vertices (SSNCV). These two latter problems were also studied by Marx in [23], where they were proven to be W[1]-hard when parameterized either by r or by w, while, when parameterized by both r and w, SSNCV admits an FPT-algorithm, while SSNV remains W[1]-hard.

CSNV was proven to be W[1]-hard when parameterized by r in [13]. Moreover, the same reduction proves that CSNCV is also W[1]-hard. Here we prove that both CSNV and CSNCV admit FPT-algorithms when parameterized by r and w. For the proof we consider an extension of List Allocation, called Bounded List Allocation (BLA). where the input additionally specifies the number of vertices that will be allocated into some of the boxes. In Subsection B.3 we give a TFTP-reduction of BLA to LA that yields a $2^{O(w^2 \cdot \log w)} \cdot n^4 \cdot \log n$-step algorithm for BLA. The CSNV problem is TFPT-reducible to BLA by a reduction that is very similar to the one that we used before for ECMC: The function λ is defined in exactly the same way, we have $r+1$ boxes, and we demand all except from the last one to receive exactly one vertex. We define $\mathcal{A}$ to contain every α where $\forall i, i', 1 \leq i < i' \leq r,\ \alpha(i, i') \in \{0, 1\}$ and $\sum_{i \in \{1,\ldots,r\}} \alpha(i, r+1) \leq w$. It is easy to see that (G, w, r) is a YES-instance of CSNV if an only if a YES-instance of BLA is generated by some $\alpha \in \mathcal{A}$. As $|\mathcal{A}| = 2^{O(r^2 + w \log r)}$ and $\sum_{1 \leq i < i' \leq r+2} \alpha = O(w + r^2)$, this reduction gives a $2^{O((r^4+w^2) \cdot \log(w \cdot r))} \cdot n^4 \cdot \log n$-step algorithm for CSNV. For for the case of CSNCV, the reduction is the same with the difference that we further restrict $\mathcal{A}$ so that the graph $(\{1, \ldots, r\}, \{\{i, i'\} \mid \alpha(i, i') = 1\})$ is connected.

List Digraph Homomorphism. Given two directed graphs G and H, an *H-homomorphism* of G is a mapping $\chi : V(G) \to V(H)$ such that if (x, y) is an arc of G, then $(\chi(x), \chi(y))$ is also an arc in H. In the List Digraph Homomorphism problem, we are given two graphs G and H and a list function $\lambda : V(G) \to 2^{V(H)}$ and we ask whether G has a (list) H-homomorphism that respects the restrictions of λ, i.e., for every vertex v of G, $\chi(v) \in \lambda(v)$. Graph homomorphisms have been extensively studied both from the combinatorial and the algorithmic point of view (see e.g., [14, 15, 20]). Especially for the List Digraph Homomorphism problem, a dichotomy characterizing the instantiations of H for which the problem is hard was given in [21] A parameterization of list homomorphism (for undirected graphs) has been introduced in [10], where the parameter is a bound on the number of pre-images of some prescribed set of vertices of H (see also [11, 24]). Another parameterization, again for the undirected case, was introduced in [7], where the parameter is the number of vertices to be removed from the graph G so that the remaining graph has a list H-homomorphism. In this paper we introduce a new, natural, parameterization where the number w of "crossing edges", i.e., the edges whose endpoints are

mapped to different vertices of H. In this paper, we prove that this parameterization of List Digraph Homomorphism can be solved in $2^{O(w^2 \cdot \log w)} \cdot n^4 \cdot \log n$ steps, where $m = |E(G)|$. The proof is again based on a TFPT-reduction to the List Allocation problem that is given in Subsection B.4.

Tree-cut width. Treewidth is a graph invariant that may serve as a measure of the topological resemblance of a graph to a tree. This invariant is of great importance for algorithmic graph theory, as a wide family of NP-hard graph problems admit FPT-algorithms when parameterized by the treewidth of their input graph. Besides that, there are interesting cases of problems where no such and FPT-algorithm is expected to exist [12, 16, 18]. Therefore, it is an interesting question whether there are different, but still general, graph invariants that can provide tractable parameterizations for such cases. The definition of several such intractable problems is critically related to edges and, intuitively, this is what makes them hard to fit in the meta-algorithmic framework that already exists for treewidth. The challenging issue here is to detect an "edge" analogue of treewidth that will provide a reasonable extension of "treewidth-based" theories of algorithm design. An interesting candidate in this direction is the graph invariant of tree-cut width that was introduced by Wollan in [30] (see Subsection B.5 for the definition). In [30] Wollan proved that *"for every planar sub-cubic graph H, there is a constant c_H such that every graph excluding H as an immersion*[2] *has bounded tree-cut width"*. The above is an analogue of the celebrated "grid exclusion theorem" (proven in [26]). In fact, the grid exclusion theorem becomes equivalent to the above statement if we replace "immersion" by "topological minor" and "tree-cut width" by "treewidth". This implies that tree-cut width has combinatorial properties analogous to those of treewidth. From the algorithmic point of view, there was some recent progress on the development of a dynamic programming framework for tree-cut width that can also work as a counterpart of the existing one on treewidth [17]. Namely, it is proved in [17] that there are problems that are W[1]-hard (or open) when parameterized by treewidth that admit FPT-algorithms when parameterized by tree-cut width. According to [17], such problems are the Capacitated Dominating Set problem, the Capacitated Vertex Cover [12], and the Balanced Vertex-Ordering problem.

A fundamental pre-requirement for the design of dynamic programming algorithms for graphs of bounded tree-cut width is to have the corresponding decomposition as part of the input. For this, it is important to *construct* an FPT-algorithm that, given a graph G and an integer w, answers correctly whether G has tree-cut width at most w and, if so, outputs an optimal tree-cut width decomposition of G. This problem was resolved for the case of treewidth by Bodlaender in [2] (see also the more recent 5-approximation FPT-algorithm in [3],

[2] A graph H is a *topological minor* (resp. *immersion*) of a graph G if H can be obtained from some subgraph of G after dissolving vertices of degree 2 (resp. lifting pairs of edges with a common endpoint). Given a graph G and two edges $e_1 = \{x, y\}$ and $e_2 = \{y, z\}$ of G, the operation of *lifting* e_1 and e_2 removes the edges e_1 and e_2 from G and adds the edge $\{x, y\}$.

with better parametric dependency). Interestingly, we know that there exists an FPT-algorithm checking whether tree-cut width at most w, because of the seminar results of [27] and [19]. However, the problem of *constructing* such an algorithm remains wide open because of the non-constructive natures of the result in [27]. In this paper we make a step towards this direction by providing a 2-approximation FPT-algorithm for this problem. In particular, we construct an algorithm that, given G and w, either outputs that the tree-cut width of G is more than w or outputs a tree-cut width decomposition of G of width at most $2w$ in $2^{O(w^4 \cdot \log w)} \cdot n^5 \cdot \log n$ steps. This algorithm is presented in Subsection B.5, makes extensive use on the combinatorial properties of tree-cut width and is essentially an TFPT-reduction to a certain graph partitioning problem that we call Special Partitioning. All the lemmata and the algorithms supporting this reduction are presented in the Appendix (Sub-subsection B.5.2).

2 Preliminaries and problem definition

2.1 Basic definitions

In this paper, when giving the running time of an algorithm of some problem whose instance involves a graph G, we agree that $n = |V(G)|$ and $m = |E(G)|$.

Functions and allocations. We use the notation $\log(n)$ to denote $\lceil \log_2(n) \rceil$ for $n \in \mathbb{Z}_{\geq 1}$ and we agree that $\log(0) = 1$. Given a non-negative integer n, we denote by $[n]$ the set of all positive integers no bigger than n. Given a finite set A and an integer $s \in \mathbb{Z}_{\geq 0}$, we denote by $\binom{A}{s}$ (resp. $\binom{A}{\leq s}$) the set of all subsets of A with exactly (resp. at most) s elements. Given two sets A and B we denote by B^A the set containing every function $f : A \to B$. Given a collection $\mathcal{F}$ of sets or graphs, we define $\bigcup \mathcal{F} = \bigcup_{S \in \mathcal{F}} S$. Given a function $h : A \to B$ and $S \subseteq A$, we define $h|_S = \{(x, y) \in h \mid x \in S\}$. Given a function $f : A \to \mathbb{Z}_{\geq 0}$ we define $\sum f = \sum_{x \in A} f(x)$. Given two functions $f_1, f_2 : A \to \mathbb{Z}_{\geq 0}$ we define $f_1 + f_2 : A \to \mathbb{Z}_{\geq 0}$ such that $(f_1 + f_2)(x) = f_1(x) + f_2(x)$. Let X be a set and let ζ_1, ζ_2 be two functions mapping X to non-negative integers. We say that $\zeta_1 \leq \zeta_2$ if $\forall i \in X,\ \zeta_1(i) \leq \zeta_2(i)$. Given a (possibly partial) function $\zeta : X \to \mathbb{Z}_{\geq 0}$ we define $\mathfrak{F}_{\leq}(\zeta) = \{\zeta' : X \to \mathbb{Z}_{\geq 0} \mid \zeta' \leq \zeta\}$ and $\mathfrak{F}^+_{\leq}(\zeta) = \mathfrak{F}_{\leq}(\zeta) \setminus \{(x, 0) \mid x \in X\}$.

An *r-allocation* of a set S is an r-tuple $\mathcal{V} = (V_1, \ldots, V_r)$ of, possibly empty, sets that are pairwise disjoint and whose union is the set S. We refer to the elements of $\mathcal{V}$ as the *parts* of $\mathcal{V}$ and we denote by $\mathcal{V}^{(i)}$ the i-th part of $\mathcal{V}$, i.e., $\mathcal{V}^{(i)} = V_i$. Given a set $T \subseteq S$, we define the *restriction of $\mathcal{V}$ to S* as the r-allocation $\mathcal{V} \cap T = (\mathcal{V}^{(1)} \cap T, \ldots, \mathcal{V}^{(r)} \cap T)$. Notice that $\mathcal{V} \cap T$ is an r-allocation of T. Given two r-allocations $\mathcal{V}_1 = (V_1^1, \ldots, V_r^1)$ and $\mathcal{V}_2 = (V_1^2, \ldots, V_r^2)$, we define $\mathcal{V}_1 \cup \mathcal{V}_2 = (V_1^1 \cup V_1^2, \ldots, V_r^1 \cup V_r^2)$.

Graphs. All graphs in this paper are loopless and they may have multiple edges. The only exception to this agreement is in Subsection B.4 where we also allow loops. Given two graphs G and G' we set $G \cup G' = (V(G) \cup V(G'), E(G) \cup E(G'))$. Given a graph G and a set $S \subseteq V(G)$, we define $\partial_G(S)$ as the set of all vertices in S that are adjacent to vertices in $V(G) \setminus S$. For a vertex set $S \subseteq V(G)$, we define $N_G(S)$ as the set of vertices in $V(G) \setminus S$ with at least one neighbor in S, and $N_G[S] = N_G(S) \cup S$.

If G is a graph and X, Y are two disjoint vertex subsets of $V(G)$, we define $\delta_G(X,Y)$ as the set of edges with one endpoint in X and the other in Y. Given a graph G denote by $\mathcal{C}(G)$ the collection of all connected components of G. Given an $S \subseteq V(G)$, we denote by $G[S]$ the subgraph of G induced by S and we also denote $G^+[S] = G[S \cup N_G(S)]$.

Let G be a connected graph. A partition (V_1, V_2) of $V(G)$ is a (q,w)-*good separation* if $|V_1|, |V_2| > q$, $|\delta_G(V_1,V_2)| \leq w$, and $G[V_1]$ and $G[V_2]$ are both connected. A graph G is called (q,w)-*connected* if it does not contain any $(q, w-1)$-*good separation*. (Note that for $q = 0$, (q,w)-connectivity corresponds exactly to classical w-edge-connectivity.)

Proposition 1 (Chitnis *et al.* [6]). *There exists a deterministic algorithm that, with input a n-vertex connected graph G, a $q \in \mathbb{Z}_{\geq 1}$ and $w \in \mathbb{Z}_{\geq 0}$, either finds a (q,w)-good separation, or reports that no such separation exists, in $2^{O(\min\{q,w\}\cdot\log(q+w))}n^3 \log n$ steps.*

2.2 The list allocation problem

Let G be a graph, $r \in \mathbb{Z}_{\geq 1}$, and let $\lambda : V(G) \to 2^{[r]}$. If H is a subgraph of G, we define the *common indices* with respect to λ of H as the set $\bigcap_{v \in V(H)} \lambda(v)$. We also say that H is (i,λ)-*friendly* if $i \in \bigcap_{v \in V(H)} \lambda(v)$.

List Allocation (LA)
Input: A tuple (G, r, λ, α) where G is a graph, $r \in \mathbb{Z}_{\geq 1}$, $\lambda : V(G) \to 2^{[r]}$, and $\alpha : \binom{[r]}{2} \to \mathbb{Z}_{\geq 0}$.
Parameter: $w = \sum \alpha$.
Output: An r-allocation $\mathcal{V}$ of $V(G)$ such that

1. $\forall \{i,j\} \in \binom{[r]}{2}$, $|\delta_G(\mathcal{V}^{(i)}, \mathcal{V}^{(j)})| = \alpha(i,j)$[3] and
2. $\forall v \in V(G), \forall i \in [r]$, if $v \in \mathcal{V}^{(i)}$ then $i \in \lambda(v)$,

or a correct report that no such r-allocation exists.

In the definition of the above problem each vertex v of G carries a *list* $\lambda(v)$ indicating the parts where v can be possibly allocated. Moreover, α is a function assigning weights to pairs of parts in $\mathcal{V}$. The weights defined by α prescribe the number of crossing edges between distinct parts of $\mathcal{V}$.

3 Description of the TFPT-reductions

In this section we give a description of the TFPT-reductions required to prove that LA admits an FPT-algorithm. Due to space restrictions, all the proofs have been moved to Appendix A.

If I is an instance of LA, we define $\mathbf{sol}(I)$ as the set of all solutions of LA for I.

Lemma 1. *There exists an algorithm that, given an instance $I = (G, r, \lambda, \alpha)$ of* LA*, computes $\mathbf{sol}(I)$ in $n^{O(w)} \cdot 2^{O((w+\ell)\cdot\log r)}$ steps, where ℓ is the number of connected components of G.*

[3] For simplicity, we write $\alpha(\{i,j\})$ as $\alpha(i,j)$ and we agree that $\alpha(i,j) = \alpha(j,i)$.

3.1 Connected list allocation

We define the Connected List Allocation problem (CLA, in short) as the List Allocation with the additional demand that the input graph is connected and $r \leq 2w$.

Lemma 2. *If there exists an algorithm solving* CLA *in* $f(w) \cdot p(n)$ *steps, then there is an algorithm that solves* LA *in* $2^{O(w^2)} \cdot f(w) \cdot p(n) + O((n+r)^2)$ *steps.*

If $I = (G, r, \lambda, \alpha)$ is an instance of CLA and $B \in \binom{V(G)}{\leq 2w}$, we set $\mathfrak{U}(I, B) = [r]^B \times \mathfrak{F}_{\leq}(\alpha)$. Let $f_1(w) = 2^w \cdot (2w)^{2w}$.

Observation 1. *For every instance* $I = (G, r, \lambda, \alpha)$ *of* CLA *and* $B \in \binom{V(G)}{\leq 2w}$, *it holds that* $|\mathfrak{U}(I, B)| \leq f_1(w)$.

Given a $\mathbf{w} = (\psi, \alpha') \in \mathfrak{U}(I, B)$, we define the instance $I_{\mathbf{w}} = (G, \lambda', r, \alpha')$ of CLA, where $\lambda' = \lambda|_{V(G)\setminus B} \cup \psi$. We also set up the function $f_2 : \mathbb{Z}_{\geq 0} \to \mathbb{Z}_{\geq 0}$ such that $f_2(w) = w \cdot (f_1(w))^2 + 2 \cdot w + 2$.

3.2 Highly connected list allocation

We define the Highly Connected List Allocation problem (HCLA, in short) as the Connected List Allocation problem with the only difference that we additionally demand that the input graph is $(f_1(w), w+1)$-connected, where w is the parameter of the problem.

Given a graph G, an integer $r \in \mathbb{Z}_{\geq 1}$, an allocation $\mathcal{V} = \{\mathcal{V}^{(1)}, \ldots, \mathcal{V}^{(r)}\}$ of $V(G)$ and two integers $j \in [r]$ and $x \in \mathbb{Z}_{\geq 0}$, we say that $\mathcal{V}$ is *x-bounded out of j* if $\sum_{i \in [r]\setminus\{j\}} |\mathcal{V}^{(i)}| \leq x$.

Lemma 3. *Let* $\mathcal{V}$ *be a solution of* HCLA *for the instance* $I = (G, r, \lambda, \alpha)$ *where* $|V(G)| \geq 2 \cdot (w+1) \cdot f_1(w)$. *Then there is a unique* $j \in [r]$ *such that* $\mathcal{V}$ *is* $w \cdot f_1(w)$*-bounded out of* j *and a unique* $C \in \mathcal{C}(G \setminus E(\mathcal{V}))$ *with* $|V(C)| > f_1(w)$. *Moreover, for such* C *and* j, C *is a subgraph of* $G[\mathcal{V}^{(j)}]$.

The definition of the set $M_{I,B}$. Let $I = (G, r, \lambda, \alpha)$ be an instance of HCLA where $|V(G)| \geq 2 \cdot (w+1) \cdot f_1(w)$ and let B be a subset of $V(G)$ of at most $2 \cdot w$ vertices. We define $\tilde{\mathfrak{U}}(I, B)$ as the set of all $\mathbf{w} \in \mathfrak{U}(I, B)$ such that $I_{\mathbf{w}}$ is a YES-instance of HCLA. We assign to each $\mathbf{w} \in \tilde{\mathfrak{U}}(I, B)$ an arbitrarily chosen solution $\mathcal{V}_{\mathbf{w}}$ for the instance $I_{\mathbf{w}}$ of HCLA. For every $\mathbf{w} \in \tilde{\mathfrak{U}}(I, B)$, we apply Lemma 3 on $I_{\mathbf{w}}$ and $\mathcal{V}_{\mathbf{w}}$ and we find the unique index $j_{\mathbf{w}} \in [r]$ such that $\mathcal{V}_{\mathbf{w}}$ is $f_1(w)$-bounded out of $j_{\mathbf{w}}$. We call the vertices in $\mathcal{V}^{(j_{\mathbf{w}})}$ $\mathbf{w}$*-marginal* and, given that $\tilde{\mathfrak{U}}(I, B) \neq \emptyset$, we define $M_{I,B} = \{v \in V(G) \mid v \text{ is } \mathbf{w}\text{-marginal for all } \mathbf{w} \in \tilde{\mathfrak{U}}(I, B)\}$.

The proof of the next lemma uses Lemma 3 and Observation 1.

Lemma 4. *If* $I = (G, r, \lambda, \alpha)$ *is a YES-instance of* HCLA *with at least* $f_2(w)$ *vertices and* B *is a subset of* $V(G)$ *of at most* $2 \cdot w$ *vertices, then* $M_{I,B} \setminus B$ *contains at least two vertices.*

The definitions of $I[Q]$ and $I\langle Q\rangle$. Let $I = (G, r, \lambda, \alpha)$ be an instance of CLA and let $Q \subseteq V(G)$. We set $I[Q] = (G[Q], r, \lambda|_Q, \alpha)$. We also set $I\langle Q\rangle = (G\langle Q\rangle, r, \lambda\langle Q\rangle, \alpha)$ where $G\langle Q\rangle$ is the result of the identification in G of all vertices of Q into a single vertex v_Q and $\lambda\langle Q\rangle = \lambda|_{V(G)\setminus Q} \cup \{(v_Q, \bigcap_{v\in Q} \lambda(v))\}$ (edges multiplicities are summed up during each identification).

The definition of $I \oplus_{\mathbf{q}} I'$. Let $I = (G, r, \lambda, \alpha)$ and $I' = (G', r, \lambda', \alpha)$ be instances of CLA, where G and G' have disjoint vertex sets. Let also $\mathbf{q} = (A, A', J)$ be a triple such that $A \subseteq V(G)$, $A' \subseteq V(G')$, and J is a set of edges, each having one endpoint in A and one endpoint in A'. We define $I \oplus_{\mathbf{q}} I' = (G \cup G' \cup (A \cup A', J), r, \lambda \cup \lambda', \alpha)$.

The proof of the next lemma uses Lemma 4.

Lemma 5. *Let $I = (G, r, \lambda, \alpha)$ be an instance of* CLA*, and let (V_1, V_2) be a bipartition of $V(G)$ such that $I[V_1]$ is an instance of* HCLA *whose graph has at least $f_2(w)$ vertices. Let also $I_i = I[V_i], i \in \{1, 2\}$, $J = \delta(V_1, V_2)$, $B_i = V_i \cap V(J)$ for $i \in \{1, 2\}$, $\mathbf{q} = (B_1, B_2, J)$, and $Q = M_{I_1,B_1} \setminus B_1$. Then I and $I' = I_1\langle Q\rangle \oplus_{\mathbf{q}} I_2$ are equivalent instances of* CLA*.*

The proof of the following lemma uses Lemmata 1 and 5.

Lemma 6. *If* HCLA *can be solved in $f(w)\cdot p(n)$ steps, then* CLA *can be solved in $\max\{2^{O(w^2\cdot\log w)} \cdot n^4 \cdot \log n, f(w) \cdot p(n) \cdot 2^{O(w\cdot\log w)}\}$ steps.*

3.3 Split highly connected list allocation

We define the Split Highly Connected List Allocation problem (SHCLA, in short) so that its instances are as the instances of Highly Connected List Allocation enhanced with some subset S of $V(G)$ and where we impose that $|V(G)| \geq 2\cdot(w+1)\cdot f_1(w)$ and that a solution $\mathcal{V}$, additionally, satisfies the following condition: There exists some $j \in [r]$, such that **A**. $\mathcal{V}$ is $w \cdot f_1(w)$-bounded out of j and **B**. $\partial_G(\mathcal{V}^{(j)}) \subseteq S \subseteq \mathcal{V}^{(j)}$. The proof of the next Lemma uses Lemma 3.

Lemma 7. *If (I, S) is a* YES*-instance of* SHCLA*, where $I = (G, r, \lambda, \alpha)$, then there exists some solution $\mathcal{V}$ of* SHCLA *for (I, S) and a unique $j \in [r]$ such that,* **i**. *$\mathcal{V}$ is $w \cdot f_1(w)$-bounded out of j and* **ii**. *if $C \in \mathcal{C}(G \setminus S)$ and C is not (j, λ)-friendly, then $V(C) \cap \mathcal{V}^{(j)} = \emptyset$.*

Proposition 2 (Chitnis *et al.* [6]). *There exists an algorithm that given a set U of size n and two integers $a, b \in [0, n]$, outputs a set $\mathcal{F} \subseteq 2^U$ with $|\mathcal{F}| = 2^{O(\min\{a,b\}\cdot\log(a+b+1))}\cdot\log n$ such that for every two sets $A, B \subseteq U$, where $A\cap B = \emptyset$ and $|A| \leq a$ and $|B| \leq b$, there exists a set $S \in \mathcal{F}$ with $A \subseteq S$ and $B \cap S = \emptyset$, in $2^{O(\min\{a,b\}\cdot\log(a+b+1))} \cdot n \cdot \log n$ steps.*

The proof of the following lemma uses Proposition 2 and Lemmata 1 and 3.

Lemma 8. *Given an algorithm solving* SHCLA *in $f(w)\cdot p(n)$ steps, then there is an algorithm solving* HCLA *in $f(w) \cdot 2^{O(w^2\cdot\log w)} \cdot \log n \cdot \max\{n, p(n)\}$ steps.*

The proof of the next lemma uses Lemmata 7 and 8.

Lemma 9. SHCLA *can be solved in in* $2^{O(w^2 \cdot \log w)} \cdot n$ *steps.*

Combining Lemmata 2, 6, 8, and 9 and the fact that $f_1(w) = 2^{O(w \cdot \log w)}$ and $f_2(w) = 2^{O(w \cdot \log w)}$ we can derive the main result of this paper.

Theorem 1. LA *can be solved in* $2^{O(w^2 \log w)} \cdot n^4 \cdot \log n$ *steps.*

4 Further research

A first natural research direction is to improve the running time of our FPT-algorithm for LA given in Theorem 1. We think that it may be a difficult task, as the dependency on the parameter, namely $2^{O(w^2 \log w)}$, is achieved at several steps of our algorithm, so in order to improve it, all these steps should be implemented more efficiently. As for the polynomial part, we think that it is somehow inherent to the technique of recursive understanding and randomized contractions, as algorithms given in [6] have also the same running time.

We believe that LA may have other applications other than the ones discussed discussed in this article. A possible extension is the Connected List Allocation problem where we additionally demand that the graphs induced by the parts of the solution to be connected and this can be treated with an easy modification of the definitions of Subsection 3.2 so that the two vertices in Lemma 4 are connected by an edge. This variant can encompass more problems such as the Edge Multiway Cut-Uncut, studied in [6]. Exploring the existence of polynomial kernels for LA is also an interesting question.

Finally, finding an explicit exact FPT-algorithm for computing the tree-cut width of a graph remains open. As the treewidth of a graph of bounded tree-cut width is also bounded [30], a possible approach would be to build a tree-cut decomposition of the input graph by performing dynamic programming over a tree decomposition of it. Nevertheless, this dynamic programming would probably be quite involved.

References

1. N. Bansal, U. Feige, R. Krauthgamer, K. Makarychev, V. Nagarajan, J. S. Naor, and R. Schwartz. Min-max graph partitioning and small set expansion. In *FOCS '11*, pp. 17–26.
2. H. L. Bodlaender. A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM J. Comput.*, 25(6):1305–1317, 1996.
3. H. L. Bodlaender, P. G. Drange, M. S. Dregi, F. V. Fomin, D. Lokshtanov, and M. Pilipczuk. An $O(c^k n)$ 5-approximation algorithm for treewidth. In *FOCS '13*, pp. 499–508.
4. J. Chen, Y. Liu, and S. Lu. An improved parameterized algorithm for the minimum node multiway cut problem. *Algorithmica*, 55(1):1–13, 2009.
5. R. Chitnis, M. Hajiaghayi, and D. Marx. Fixed-parameter tractability of directed multiway cut parameterized by the size of the cutset. In *SODA '12*, pp. 1713–1725.
6. R. H. Chitnis, M. Cygan, M. Hajiaghayi, M. Pilipczuk, and M. Pilipczuk. Designing FPT algorithms for cut problems using randomized contractions. In *FOCS 2012*, pp. 460–469.

7. R. H. Chitnis, L. Egri, and D. Marx. List h-coloring a graph by removing few vertices. In *ESA 2013*, pp. 313–324.
8. M. Cygan, D. Lokshtanov, M. Pilipczuk, M. Pilipczuk, and S. Saurabh. Minimum bisection is fixed parameter tractable. In *STOC '14*, pp. 323–332.
9. E. Dahlhaus, D. S. Johnson, C. H. Papadimitriou, P. D. Seymour, and M. Yannakakis. The complexity of multiterminal cuts. *SIAM J. Comput.*, 23(4):864–894, Aug. 1994.
10. J. Díaz, M. Serna, and D. M. Thilikos. (H, C, K)-coloring: fast, easy, and hard cases. In MFCS 2001, pp. 304–315.
11. J. Díaz, M. Serna, and D. M. Thilikos. Efficient algorithms for counting parameterized list H-colorings. *J. Comput. System Sci.*, 74(5):919–937, 2008.
12. M. Dom, D. Lokshtanov, S. Saurabh, and Y. Villanger. Capacitated domination and covering: A parameterized perspective. In *IPEC 2008*, pp. 78–90.
13. R. G. Downey, V. Estivill-Castro, M. Fellows, E. Prieto, and F. A. Rosamund. Cutting up is hard to do: The parameterised complexity of k-cut and related problems. *Electronic Notes in Theoretical Computer Science*, 78(0):209 – 222, 2003.
14. L. Egri, A. Krokhin, B. Larose, and P. Tesson. The complexity of the list homomorphism problem for graphs. *Theor. of Comp. Syst.*, 51(2):143–178, 2012.
15. T. Feder, P. Hell, and J. Huang. Bi-arc graphs and the complexity of list homomorphisms. *Journal of Graph Theory*, 42(1):61–80, 2003.
16. M. R. Fellows, F. V. Fomin, D. Lokshtanov, F. Rosamond, S. Saurabh, S. Szeider, and C. Thomassen. On the complexity of some colorful problems parameterized by treewidth. *Information and Computation*, 209(2):143 – 153, 2011.
17. R. Ganian, E. Kim, and S. Szeider. Algorithmic applications of tree-cut width. Manuscript, 2014.
18. P. A. Golovach and D. M. Thilikos. Paths of bounded length and their cuts: Parameterized complexity and algorithms. *Discrete Optimization*, 8(1):72–86, 2011.
19. M. Grohe, K. Kawarabayashi, D. Marx, and P. Wollan. Finding topological subgraphs is fixed-parameter tractable. In *STOC 2011*, pp. 479–488.
20. P. Hell and J. Nešetřil. *Graphs and homomorphisms*, volume 28 of *Oxford Lecture Series in Mathematics and its Applications*. Oxford University Press, Oxford, 2004.
21. P. Hell and A. Rafiey. The dichotomy of list homomorphisms for digraphs. In *SODA 2011*, pp. 1703–1713.
22. L. M. Kirousis, M. Serna, and P. Spirakis. Parallel complexity of the connected subgraph problem. *SIAM J. Comput.*, 22(3):573–586, 1993.
23. D. Marx. Parameterized graph separation problems. *Theoretical Computer Science*, 351(3):394 – 406, 2006. Parameterized and Exact Computation 2004.
24. D. Marx, B. O'Sullivan, and I. Razgon. Finding small separators in linear time via treewidth reduction. *ACM Transactions on Algorithms*, 9(4):30, 2013.
25. H. Räcke. Optimal hierarchical decompositions for congestion minimization in networks. In *STOC '08*, pp. 255–264.
26. N. Robertson and P. D. Seymour. Graph Minors. V. Excluding a planar graph. *Journal of Combinatorial Theory, Series B*, 41(2):92–114, 1986.
27. N. Robertson and P. D. Seymour. Graph minors XXIII. Nash-Williams' immersion conjecture. *J. Combin. Theory Ser. B*, 100(2):181–205, 2010.
28. M. Stoer and F. Wagner. A simple min-cut algorithm. *Journal of the ACM*, 44(4):585–591, 1997.
29. Z. Svitkina and E. Tardos. Min-max multiway cut. In K. Jansen, S. Khanna, J. Rolim, and D. Ron, editors, *APPROX/RANDOM 2004*, pp. 207–218.
30. P. Wollan. The structure of graphs not admitting a fixed immersion. *J. Comb. Theory, Ser. B*, 110:47–66, 2015.

A Proofs of Section 3

A.1 Proof of Lemma 1

Given an instance $I = (G, r, \lambda, \alpha)$ of LA, we say that a pair $\{i, j\} \in \binom{[r]}{2}$ is *a positive pair of* I if $\alpha(i, j) > 0$. An index $i \in [r]$ is a *positive index* of I if it belongs to some positive pair of I. We use notation $[r]^*$ to denote the positive indices of I. In what follows, we always assume that positive indices of each instance I of LA form a prefix of $[r]$, otherwise we rearrange the indices so that this is the case.

If I is an instance of LA, we define $E(\mathcal{V}) = \bigcup_{\{i,j\} \in \binom{[r]}{2}} \delta_G(\mathcal{V}^{(i)}, \mathcal{V}^{(j)})$.

Observation 2. *If $\mathcal{V}$ is a solution for some instance $I = (G, r, \lambda, \alpha)$ of* LA *then every connected component of $G \setminus E(\mathcal{V})$ is also a connected component of $G[\mathcal{V}^{(i)}]$ for some $i \in [r]$.*

Observation 3. *If $\mathcal{V}$ is a solution for an instance $I = (G, r, \lambda, \alpha)$ of* LA*, where G has ℓ connected components, then $G \setminus E(\mathcal{V})$ contains at most $w + \ell$ connected components.*

Proof:[of Lemma 1] The algorithm considers each subset F of $E(G)$ of size w. Notice that there are $n^{O(w)}$ such subsets. From Observation 3, $G_F = G \setminus F$ has at most $w + \ell$ connected components. From Observation 2, if $\mathcal{V}$ is a solution of LA for I, and $E(\mathcal{V}) = F$, then the vertex set of each connected component of G_F is entirely contained in some $\mathcal{V}^{(i)}$. The algorithm considers all possible ways to assign the $\leq w + \ell$ connected components of G_F to the r indices of I and checks whether this creates a solution for I. As there are $2^{O((w+\ell)\cdot \log r)}$ such assignments, the claimed running time follows. □

A.2 Proof of Lemma 2

Given an instance $I = (G, r, \lambda, \alpha)$ of CLA, we define

$$\mathbf{folio}(I) = \{\alpha' \in \mathfrak{F}_{\leq}(\alpha) \mid I = (G, r, \lambda, \alpha') \text{ is a YES-instance of CLA}\}.$$

Let $\mathcal{W}$ be a collection of connected subgraphs of G and let $\lambda : V(G) \to [r]$. For each $\alpha' \in \mathfrak{F}_{\leq}(\alpha)$ we define

$$\mathbf{rep}(\mathcal{W}, \alpha') = \{C \in \mathcal{W} \mid \alpha' \in \mathbf{folio}(C, r, \lambda|_{V(C)}, \alpha)\}$$

and, for every set $S \subseteq V(G)$, we define

$$\mathbf{trunk}(I, \mathcal{W}, S) = \bigcup_{\alpha' \in \mathfrak{F}_{\leq}(\alpha)} \tilde{\mathbf{rep}}(\mathcal{W}, \alpha')$$

where, for each $\alpha' \in \mathfrak{F}_{\leq}(\alpha)$, $\tilde{\mathbf{rep}}(\mathcal{W}, \alpha')$ consists of $\min\{w, |\mathbf{rep}(\mathcal{W}, \alpha')|\}$ smallest, with respect to the number of vertices not in S, elements of $\mathbf{rep}(\mathcal{W}, \alpha')$.

Observation 4. *Given an instance I of* CLA*, and a collection $\mathcal{W}$ of connected subgraphs of G, and a set $S \subseteq V(G)$, the set* $\mathbf{trunk}(I, \mathcal{W}, S)$ *contains at most $w \cdot 2^w$ elements and can be computed in $O(2^{O(w)} \cdot n)$ steps. Moreover, for every $\alpha' \in \mathfrak{F}_{\leq}(\alpha^*)$, $|\mathbf{trunk}(I, \mathcal{W}, S) \cap \mathbf{rep}(\mathcal{W}, \alpha')| \geq \min\{w, |\mathbf{rep}(\mathcal{W}, \alpha')\}$.*

Observation 5. *Each instance I of* LA *has at most w positive pairs and $2w$ positive indices.*

Proof:[of Lemma 2] Suppose that A is an algorithm that solves CLA in $f(w) \cdot p(n)$ steps.

Let $r^* = 2w$, $\alpha^* = \alpha|_{\binom{[r^*]}{2}}$, and $\lambda^* = \lambda|_{2^{[r^*]}}$. Let also $I^* = (G, r^*, \lambda^*, \alpha^*)$. We define

$$\mathcal{W} = \{C \in \mathcal{C}(G) \mid C \text{ is } (j, \lambda)\text{-friendly for some } j \in [r]\},$$

and set $\mathcal{Y} = \mathcal{C}(G) \setminus \mathcal{W}$. Observe that $\mathcal{Y}$ can be computed in $O((n + r)^2)$ steps. Notice also that if I is a YES-instance of LA, then $\mathcal{Y}$ contains at most w connected components of G. From now on we assume that $|\mathcal{Y}| \leq w$. Let $\tilde{\mathcal{W}} = \mathbf{trunk}(I^*, \mathcal{W}, \emptyset)$.

Claim 1. *I is a YES-instance of* LA *if and only if, for some $\mathcal{Q} \in \binom{\tilde{\mathcal{W}}}{\leq w}$, $(\tilde{G}_{\mathcal{Q}}, r^*, \lambda^*, \alpha^*)$ is a YES-instance of* LA *where $\tilde{G}_{\mathcal{Q}} = G[\mathbf{U}\mathcal{Q} \cup \mathbf{U}\mathcal{Y}]$.*

Proof of the Claim: Let $\mathcal{V}^*$ be a solution for the instance $(\tilde{G}_{\mathcal{Q}}, r^*, \lambda^*, \alpha^*)$ for some $\mathcal{Q} \in \binom{\tilde{\mathcal{W}}}{\leq w}$. Recall that for every $C \in \mathcal{W} \setminus \mathcal{Q}$ there exists some $j_C \in [r]$ such that C is (j_C, λ)-friendly. We define the r-allocation $\mathcal{V}$ such that, for $i \in [r]$, $\mathcal{V}^{(i)} = \mathcal{V}^{*(i)} \cup \{V(C) \mid j_C = i\}$ and observe that $\mathcal{V}$ is a solution of LA for I.

Assume now that $\mathcal{V}$ is a solution of LA for I. Given such a solution we define the set

$$\mathcal{Q}_{\mathcal{V}} = \{C \in \mathcal{W} \mid \forall i \in [r], V(C) \not\subseteq \mathcal{V}^{(i)}\}$$

and we choose I such that the quantity $|\mathcal{Q}_{\mathcal{V}} \setminus \tilde{\mathcal{W}}|$ is minimized.

Notice that $|\mathcal{Q}_{\mathcal{V}}| \leq w$. Our first step is to prove that $\mathcal{Q}_{\mathcal{V}} \subseteq \tilde{\mathcal{W}}$. Suppose on the contrary that C is a connected component in $\mathcal{Q}_{\mathcal{V}}$ that does not belong in $\tilde{\mathcal{W}}$. Let $\alpha_C : \binom{[r]}{2} \to [r]$ such that

$$\forall \{i, j\} \in \binom{[r]}{2},\ \alpha_C(i, j) = |\delta_G(\mathcal{V}^{(i)} \cap V(C), \mathcal{V}^{(j)} \cap V(C))|.$$

Notice that $C \in \mathbf{rep}(\mathcal{W}, \alpha_C)$. As $C \notin \tilde{\mathcal{W}}$, we obtain that $|\mathbf{rep}(\mathcal{W}, \alpha_C)| \geq w$, therefore $|\tilde{\mathcal{W}} \cap \mathbf{rep}(\mathcal{W}, \alpha_C)| \geq w$. This in turn implies that, among the connected components in $\tilde{\mathcal{W}} \cap \mathbf{rep}(\mathcal{W}, \alpha_C)$, at least one, say C', of them does not belong to $\mathcal{Q}_{\mathcal{V}} \setminus \{C\}$.

As $C' \in \mathbf{rep}(\mathcal{W}, \alpha_C)$ we have that $(C', r, \lambda|_{V(C')}, \alpha_C)$ is a YES-instance of CLA and let $\mathcal{V}_{C'}$ be a solution for this instance. Let also $j_C \in [r]$ such that C is (j_C, λ)-friendly (we know that this is the case because $C \in \mathcal{Q}_{\mathcal{V}} \subseteq \mathcal{W}$). Let $\mathcal{V}_C$ be an r-allocation of $V(C)$ such that $\mathcal{V}_C^{(j_C)} = V(C)$ and $\mathcal{V}_C^{(i)} = \emptyset$ for $i \in [r] \setminus \{j_C\}$. We now set $\mathcal{V}' = \mathcal{V}'' \cup \mathcal{V}_{C'} \cup \mathcal{V}_C$, where $\mathcal{V}'' = \mathcal{V} \cap (V(G) \setminus (V(C) \cup V(C')))$.

Observe that $\mathcal{V}'$ is a solution of CLA for I. Notice that $\mathcal{Q}_{\mathcal{V}'}$ has one more element from $\tilde{W}$ than $\mathcal{Q}_{\mathcal{V}}$ (i.e., the connected component C'). This means that $|\mathcal{Q}_{\mathcal{V}'} \setminus \tilde{\mathcal{W}}| < |\mathcal{Q}_{\mathcal{V}} \setminus \tilde{\mathcal{W}}|$, a contradiction to the choice of $\mathcal{V}$. This completes the proof that $|\mathcal{Q}_{\mathcal{V}}| \leq w$.

Let $\mathcal{Q} = \mathcal{Q}_{\mathcal{V}}$. It now remains to verify that $I^* = (\tilde{G}_{\mathcal{Q}}, r^*, \lambda^*, \alpha^*)$ is a YES-instance of LA. For this it is enough to observe that $\mathcal{V} \cap V(\tilde{G}_{\mathcal{Q}})$ is a solution of CLA for I^* and this completes the proof of the claim. $\diamond$

Notice that $|\binom{\tilde{\mathcal{W}}}{\leq w}| \leq 2^{O(w^2)}$. It remains to give an algorithm for checking whether $(\tilde{G}_{\mathcal{Q}}, r^*, \lambda^*, \alpha^*)$ is a YES-instance of LA, given that $\tilde{G}_{\mathcal{Q}}$ has at most $2w$ connected components. For this, we fix an ordering $C_1, \ldots, C_s$ of the members of $\mathcal{Q} \cup \mathcal{Y}$, and for every sequence $\alpha_1, \ldots, \alpha_s \in \mathfrak{F}^+_{\leq}(\alpha^*)$ where $\sum_{i \in [s]} \alpha_i = \alpha$, we use algorithm A to check whether, for $i \in [s]$, $(C_i, r^*, \lambda^*, \alpha_i)$ is a YES-instance of CLA. Notice that there are $2^{O(w \cdot \log w)}$ choices for the sequence $\alpha_1, \ldots, \alpha_s$. This yields the claimed running time. $\square$

A.3 Proof of Lemma 3

Proof:[of Lemma 3] Let C be a connected component of $G \setminus E(\mathcal{V})$ that has maximum number of vertices. As $G \setminus E(\mathcal{V})$ has at most $w+1$ connected components and $2 \cdot (w+1) \cdot f_1(w) \geq (w+1) \cdot f_1(w) + 1$, we deduce that $|V(C)| > f_1(w)$. Using Observation 2, we know that C belongs entirely in some $\mathcal{V}^{(j)}$. As G is $(f_1(w), w+1)$-connected, every connected component of $G \setminus E(\mathcal{V})$ that is different from C has at most $f_1(w)$ vertices. This implies that the union of the parts of $\mathcal{V}$ that are different from $\mathcal{V}^{(j)}$ contains at most $w \cdot f_1(w)$ vertices. Moreover j is unique as, otherwise, $|V(G)| \leq 2 \cdot w \cdot f_1(w)$. $\square$

A.4 Proof of Lemma 4

Proof:[of Lemma 4] Let $\mathbf{w} \in \mathfrak{U}(I, B)$. Since G is $(f_1(w), w+1)$-connected and $f_2(w) \geq 2 \cdot (w+1) \cdot f_1(w)$, Lemma 3 implies that there are at most $w \cdot f_1(w)$ vertices in $V(G)$ that are not $\mathbf{w}$-marginal. This, together with Observation 1, implies that there are at most $w \cdot (f_1(w))^2$ vertices in $V(G)$ that do not belong in $M_{I,B}$. The proof of the lemma follows as $|V(G)| \geq w \cdot (f_1(w))^2 + 2 \cdot w + 2$ and $|B| \leq 2 \cdot w$. $\square$

A.5 Proof of Lemma 5

Proof:[of Lemma 5] We need to prove that I is a YES-instance of CLA if and only if I' is. Note that by Lemma 4 it holds that $|Q| \geq 2$, and therefore the instance $I_1\langle Q \rangle$ is well-defined. Note also that I' is indeed an instance of CLA, as the identification of the vertices in Q does not break the connectivity of G, and r and α are the same in I and in I', so $r \leq 2 \sum \alpha$ holds. Let V_1' be the vertex set resulting from V_1 after identifying all vertices of Q into a vertex v_Q

(recall that edge multiplicities are summed up after the identification), and let G' be the graph of I'.

Assume first that I is a YES-instance, and let $\mathcal{V}$ be a solution of CLA for I. Let $\psi_1 = \{(v, \mathcal{V}(v)) \mid v \in B_1\}$, where $\mathcal{V}(v)$ denotes the integer $i \in [r]$ such that $v \in \mathcal{V}^{(i)}$. Let also α_1 be the element of $\mathfrak{F}_{\leq}(\alpha)$ such that for any two distinct integers $i, j \in [r]$, $\alpha_1(i,j) = |\delta_{G[V_1]}(\mathcal{V}^{(i)}, \mathcal{V}^{(j)})|$, and let $\mathbf{w}_1 = (\psi_1, \alpha_1)$. Note that by the definition of $\mathbf{w}_1$ and since we assume that $I[V_1]$ is an instance of HCLA, it holds that $\mathbf{w}_1 \in \tilde{\mathfrak{U}}(I_1, B_1)$. Since Q contains only vertices of V_1 that are $\mathbf{w}$-marginal for all $\mathbf{w} \in \tilde{\mathfrak{U}}(I_1, B_1)$, in particular for $\mathbf{w}_1$, there exists a solution $\mathcal{V}_{\mathbf{w}_1}$ for the instance $I_{\mathbf{w}_1}$ of HCLA such that all the vertices in Q belong to the same part of $\mathcal{V}_{\mathbf{w}_1}$, say $\mathcal{V}_{\mathbf{w}_1}^{(j_{\mathbf{w}_1})}$. Let $\mathcal{V}_{v_Q}$ be the r-allocation of the vertex set $\{v_Q\}$ such that $\mathcal{V}_{v_Q}^{(j_{\mathbf{w}_1})} = \{v_Q\}$ and for all $i \in [r], i \neq j_{\mathbf{w}_1}$, $\mathcal{V}_{v_Q}^{(i)} = \emptyset$. It follows that the r-allocation

$$(\mathcal{V}_{\mathbf{w}_1} \cap (V_1 \setminus Q)) \cup \mathcal{V}_{v_Q} \cup (\mathcal{V} \cap V_2)$$

is a solution of CLA for I'.

Conversely, assume now that I' is a YES-instance. Let $\mathcal{V}'$ be a solution of CLA for I', let $j_Q \in [r]$ be such that $v_Q \in \mathcal{V}'^{(j_Q)}$, and let $\mathcal{V}_Q$ be the r-allocation of the vertex set Q such that $\mathcal{V}_Q^{(j_Q)} = Q$ and for all $i \in [r], i \neq j_Q$, $\mathcal{V}_Q^{(i)} = \emptyset$. We claim that the r-allocation

$$\mathcal{V} = (\mathcal{V}' \cap (V_1' \setminus \{v_Q\})) \cup \mathcal{V}_Q \cup (\mathcal{V}' \cap V_2)$$

of $V(G)$ is a solution of CLA for I. Indeed, since $\mathcal{V}'$ is a solution of CLA for I', in particular we have that $\bigcap_{v \in Q} \lambda(v) \neq \emptyset$, and therefore for all $v \in Q$, if $v \in \mathcal{V}^{(i)}$ then $i \in \lambda(v)$. On the other hand, since the only change in $\mathcal{V}$ with respect to $\mathcal{V}'$ are the vertices of Q, which all belong to part $\mathcal{V}^{(j_Q)}$, it holds that for any two distinct integers $i, j \in [r]$, $|\delta_G(\mathcal{V}^{(i)}, \mathcal{V}^{(j)})| = |\delta_{G'}(\mathcal{V}'^{(i)}, \mathcal{V}'^{(j)})| = \alpha(i,j)$. □

A.6 Proof of Lemma 6

Observation 6. *If there exists an algorithm that can find, if it exists, a solution of* HCLA *in $f(w) \cdot p(n)$ steps, then there is an algorithm that, given an instance $I = (G, r, \lambda, \alpha)$ of* HCLA *and a set $B \subseteq V(G)$ where $|B| \leq 2 \cdot w$, computes the set $\tilde{\mathfrak{U}}(I, B)$, the set $\{\mathcal{V}_{\mathbf{w}} \mid \mathbf{w} \in \tilde{\mathfrak{U}}(I, B)\}$, and (in case $\tilde{\mathfrak{U}}(I, B) \neq \emptyset$) the set $M_{I,B}$ in $f(w) \cdot p(n) \cdot f_1(w)$ steps.*

Proof:[of Lemma 6] Let $I = (G, r, \lambda, \alpha)$ be an instance of CLA. If G has less than $f_2(w)$ vertices then, because of Lemma 1 and the fact that $r \leq 2w$, the problem can be solved in $2^{O(w^2 \cdot \log(w)}$ steps. If not, we apply the same Lemma on **shrink**$(I, \emptyset)$. The correctness of Algorithm **shrink** follows immediately from Lemma 5. Therefore what remains is to prove that it runs in the claimed running time, assuming that there exists an algorithm that solves HCLA in $f(w) \cdot p(n)$ steps.

Let now $T(n, w)$ be the running time of Algorithm **shrink** when it runs on an instance $I = (G, r, \lambda, \alpha)$ where $|V(G)| = n$ and $w = \sum \alpha$. Notice that

Algorithm: **shrink**(I, B)
Input : An instance $I = (G, r, \lambda, \alpha)$ of CLA and a set $B \subseteq V(G)$ where $|B| \leq 2 \cdot w$ and $|V(G)| \geq f_2(w)$.
Output : An instance I^{new} that is equivalent to I whose graph has less than $f_2(w)$ vertices or a report that I is a NO-instance.

1. **if** *G has a $(f_1(w), w+1)$-separation (V_1, V_2)* **then**
2. **let** i be an integer in $\{1, 2\}$ such that $|B \cap V_i| \leq w$
3. **let** $J = \delta(V_1, V_2)$, $A_i = V_i \cap V(J)$, and $A_{3-i} = V_{3-i} \cap V(J)$
4. **let** $B' = (B \cap V_i) \cup A_i$
5. **let** $I' = I[V_i]$ and $I'' = I[V_{3-i}]$
6. **let** I'^{new}= **shrink**(I', B')
7. **let** $I^{\text{new}} = I'^{\text{new}} \oplus_{\mathbf{q}} I''$, where $\mathbf{q} = (A_1, A_2, J)$
8. **if** $|V(I^{\text{new}})| \geq f_2(w)$ **then**
9. **return** **shrink**(I^{new}, B)
10. **else**
11. **return** I^{new}
12. **end**
13. **else**
14. **if** $\tilde{\mathfrak{U}}(I, B) = \emptyset$ **then**
15. report that I is a NO-instance
16. **else**
17. compute $M_{I,B}$
18. **let** $Q = M_{I,B} \setminus B$
19. **return** $I^{\text{new}} = I\langle Q \rangle$
20. **end**
21. **end**

$$T(n, w) \leq \max_{f_1(w) \leq n' \leq n - f_1(w)} \{T_1(n, w) + T(n', w) + T(f_2(w) + n - n', w), T_2(n, w)\},$$

where T_1 is the running time of required by line **1** and T_2 is the running time required to compute $M_{I,B}$ in line **17**. From Proposition 1, $T_1(n, w) = 2^{O(w^2 \cdot \log w)} \cdot n^3 \cdot \log n$ and, from Observation 6, $T_2(n, w) = f(w) \cdot p(n) \cdot f_1(w)$. By resolving the above recursion, we obtain that $T(n, w) = \max\{T_1(n, w) \cdot n, T_2(n, w)\}$, which yields the claimed running time. □

A.7 Proof of Lemma 7

Proof:[of Lemma 7] Let $\mathcal{V}$ be a solution of SHCLA for (I, S). From Lemma 3, there is a unique index j satisfying **A**. We also adjust the choice of $\mathcal{V}$ such that $|\mathcal{V}^{(j)}|$ is maximized. Obviously Condition **i** holds. To prove Condition **ii**, consider a connected component C of $G \setminus S$. Suppose that C is not (j, λ)-friendly. Clearly, $V(C)$ is not a subset of $\mathcal{V}^{(j)}$, therefore it contains some vertex x not in $\mathcal{V}^{(j)}$. Towards a contradiction we assume that C has also a vertex y in $\mathcal{V}^{(j)}$. This is

impossible as every path between x and y in C should contain some vertex z of $\partial_G(\mathcal{V}^{(j)})$. From **B**, $z \in S$, a contradiction as C is a connected component of $G \setminus S$. □

A.8 Proof of Lemma 8

Proof:[of Lemma 8] Let I be an instance of HCLA. If $|V(G)| < 2\cdot(w+1)\cdot f_1(w)$, HCLA can be solved in $(2 \cdot (w+1) \cdot f_1(w))^w \cdot 2^{O(w \cdot \log w)} = 2^{O(w^2 \cdot \log w)}$ steps because of Lemma 1 (applied for $\ell = 1$).

Let $\mathcal{F}$ be a family of subsets of $V(G)$ such that the condition of Proposition 2 is satisfied for $a = w$ and $b = w \cdot f_1(w)$. We claim that I is a YES-instance of HCLA if and only if for some $S \in \mathcal{F}$, (I, S) is a YES-instance of SHCLA. Recall that (I, S) is an instance of SHCLA, as $|V(G)| \geq 2 \cdot (w+1) \cdot f_1(w)$.

In the non-trivial direction, assume that $\mathcal{V}$ is a solution for I. By applying Lemma 3 on I, we know that there is a unique j such that $\mathcal{V}$ is $w \cdot f_1(w)$-bounded out of j. Let $A = \partial_G(\mathcal{V}^{(j)})$ and $B = \bigcup_{i \in [r] \setminus \{j\}} \mathcal{V}^{(i)}$. Clearly, $|A| \leq w$ and $|B| \leq w \cdot f_1(w)$. By the definition of $\mathcal{F}$, there exists some set $S \in \mathcal{F}$ such that $A \subseteq S$ and $B \cap S = \emptyset$. Therefore $\partial_G(\mathcal{V}^{(j)}) \subseteq S \subseteq \mathcal{V}^{(j)}$ and (I, S) is a YES-instance of SHCLA as required.

Suppose now that A is an algorithm that solves SHCLA in $f(w) \cdot p(n)$ steps. To solve HCLA, we apply A on (I, S) for all $S \in \mathcal{F}$. If we obtain a solution to (I, S) for some $S \in \mathcal{F}$ we output this solution as a solution to I, otherwise we output that I is a NO-instance of HCLA. As $|\mathcal{F}| = 2^{O(w \cdot \log(w \cdot f_1(w)))} \cdot \log n = 2^{O(w^2 \cdot \log w)} \cdot \log n$, this algorithm runs in $2^{O(w^2 \cdot \log w)} \cdot \log n \cdot n + 2^{O(w^2 \cdot \log w)} \cdot \log n \cdot f(w) \cdot p(n)$ steps as required. □

A.9 Proof of Lemma 9

Let (I, S) be an instance of SHCLA where $I = (G, r, \lambda, \alpha)$ and $S \subseteq V(G)$, and let $\mathcal{W} \subseteq \mathcal{C}(G \setminus S)$ and $j \in [r]$. We define the function $\lambda_{j,S} : V(G) \to 2^{[r]}$ such that

$$\lambda_{j,S}(x) = \begin{cases} \{j\} & x \in S, \\ \lambda(x) \setminus \{j\} & x \in V(G) \setminus S. \end{cases}$$

Proof:[of Lemma 9] Our first aim is to prove that if **shcla-solver**(I, S) outputs some r-allocation $\mathcal{V}$ of $V(G)$ then $\mathcal{V}$ is a solution of SHCLA for (I, S). By Line **17**, $\mathcal{V}$ was produced by some r-allocation $\mathcal{V}^*$ of $V(G_{\mathcal{Q}})$ where $\mathcal{V}^*$ is a solution to $(G_{\mathcal{Q}}, r, \lambda_{j,S}|_Q, \alpha)$, for some choice of j and $\mathcal{Q}$ and $\mathcal{V}^*$ is $w \cdot f_1(w)$-bounded out of j.

As $j \in L = \bigcap_{v \in S} \lambda(v)$, it follows that for every $x \in Q$, $\lambda_{j,S}|_Q(x) \subseteq \lambda|_Q(x)$. This implies that $\mathcal{V}^*$ is a solution of LA for $I = (G_{\mathcal{Q}}, r, \lambda|_Q, \alpha)$. Moreover, by the definition of $\lambda_{j,S}$, $S \cap Q = \mathcal{V}^{*(j)}$. This implies that

$$\partial_{G_{\mathcal{Q}}}(\mathcal{V}^{*(j)}) \subseteq \mathcal{V}^{*(j)} \subseteq S. \tag{1}$$

Algorithm: **shcla-solver**(I, S)
Input : An instance (I, S) of SHCLA, where $I = (G, r, \lambda, \alpha)$.
Output : A solution $\mathcal{V}$ of SHCLA for (I, S) or a report that (I, S) is a NO-instance.

let $L = \bigcap_{v \in S} \lambda(v)$
if $L = \emptyset$ **then return** that (I, S) is a NO-instance
for $j \in L$
do
  let $\{\mathcal{W}, \mathcal{Y}, \mathcal{Z}\}$ be a partition of $\mathcal{C}(G \setminus S)$ such that
    $\mathcal{W}$ contains every (j, λ)-friendly graph $C \in \mathcal{C}(G \setminus S)$ where
      $|V(C)| \leq w \cdot f_1(w)$ and $|N_G(V(C))| \leq w$, and
    $\mathcal{Y}$ is the set of all graphs in $\mathcal{C}(G \setminus S)$ that are not (j, λ)-friendly.
  let $\mathcal{W}^+ = \{G^+[V(C)] \mid C \in \mathcal{W}\}$ and $\mathcal{Y}^+ = \{G^+[V(C)] \mid C \in \mathcal{Y}\}$
  let $I_j = (G, r, \lambda_{j,S}, \alpha)$ and $\tilde{\mathcal{W}}^+ = \textbf{trunk}(I_j, \mathcal{W}^+, S)$
  for $\mathcal{Q} \in \binom{\mathcal{Y}^+ \cup \tilde{\mathcal{W}}^+}{\leq w}$ such that $\mathcal{Y}^+ \subseteq \mathcal{Q}$ **do**
    let $Q = V(\bigcup \mathcal{Q})$, $G_{\mathcal{Q}} = G[Q]$, and
    if $|Q| \leq w \cdot (f_1(w) + 1)$ and there exists a $\mathcal{V}^* \in \textbf{sol}(G_{\mathcal{Q}}, r, \lambda_{j,S}|_Q, \alpha)$
      such that $\mathcal{V}^*$ is $w \cdot f_1(w)$-bounded out of j, **then**
        let $\mathcal{V}$ be the r-allocation obtained from $\mathcal{V}^*$ by adding in $\mathcal{V}^{*(j)}$ the vertices in S together
          with the vertices of all the graphs in $\mathcal{C}(G \setminus S)$ except from those that intersect Q
      return $\mathcal{V}$
  end
end
return that (I, S) is a NO-instance

Notice now that $\mathcal{V}$ is created by adding in the j-part of $\mathcal{V}^*$ the vertices of S together with the vertices of all the connected components of $\mathcal{C}(G \setminus S)$ except from those intersecting Q. Clearly, $S \subseteq \mathcal{V}^{(j)}$.

Claim 2. $N_G(Q \setminus \mathcal{V}^{*(j)}) \subseteq \mathcal{V}^{*(j)}$.

Proof of the Claim: Let $\{Q_1, Q_2\}$ be the partition of Q where Q_1 contains the vertices that belong to S and Q_2 contains the vertices that belong to some connected component of $\mathcal{C}(G \setminus S)$. As $S \cap Q = \mathcal{V}^{*(j)}$, we obtain that $Q_1 = \mathcal{V}^{*(j)}$ and as $\mathcal{V}^*$ is an r-alocation of Q, we also have that $Q_2 = \bigcup_{i \in [r] \setminus \{j\}} \mathcal{V}^{*(i)}$. From the construction of $\mathcal{W}^+$ and $\mathcal{Y}^+$ we have that every neighbor of a vertex in Q_2 in G is a member of Q, therefore $N_G(Q_2) \subseteq Q_1$ and the claim holds. $\diamond$

Claim 3. $\partial_G(\mathcal{V}^{(j)}) = \partial_{G_\mathcal{Q}}(\mathcal{V}^{*(j)})$.

Proof of the Claim: We prove that $\partial_G(\mathcal{V}^{(j)}) \subseteq \partial_{G_\mathcal{Q}}(\mathcal{V}^{*(j)})$ as the other direction is trivial. Let $x \in \partial_G(\mathcal{V}^{(j)})$. Then there exists a vertex $y \in V(G) \setminus \mathcal{V}^{(j)}$ such that $\{x, y\} \in E(G)$. By the construction of $\mathcal{V}$, we have that $V(G) \setminus \mathcal{V}^{(j)} = Q \setminus \mathcal{V}^{*(j)}$ and thus $y \in Q \setminus \mathcal{V}^{*(j)}$. As $x \in \partial_G(\mathcal{V}^{(j)}) \subseteq \mathcal{V}^{(j)}$ and $\mathcal{V}$ is an r-allocation, we have that $x \notin V(G) \setminus \mathcal{V}^j = Q \setminus \mathcal{V}^j$. Hence $x \in N_G(Q \setminus \mathcal{V}^{*(j)})$ which, from Claim 2 implies that $x \in \mathcal{V}^{*(j)}$. Observe that both x and y are vertices of Q, therefore $\{x, y\} \in E(G_\mathcal{Q})$. It follows that $x \in \partial_{G_\mathcal{Q}}(\mathcal{V}^{*(j)})$. $\diamond$

A direct consequence of the above claim and (1) is that $\partial_G(\mathcal{V}^{(j)}) \subseteq S \subseteq \mathcal{V}^{(j)}$. As $Q \setminus \mathcal{V}^{*(j)} = V(G) \setminus \mathcal{V}^{(j)}$ and $\mathcal{V}^*$ is $w \cdot f_1(w)$-bounded out of j it also follows that $\mathcal{V}$ is $w \cdot f_1(w)$-bounded out of j. Therefore $\mathcal{V}$ satisfies conditions **A** and **B**.

It remains to show that $\mathcal{V}$ is a solution of LA for I. For Condition 1 notice that $Q \setminus \mathcal{V}^{*(j)} = V(G) \setminus \mathcal{V}^{(j)}$ implies that $|\delta_G(\mathcal{V}^{(i)}, \mathcal{V}^{(w)})| = |\delta_{G_\mathcal{Q}}(\mathcal{V}^{*(i)}, \mathcal{V}^{*(w)})| = \alpha(i, w)$, for $\{i, w\} \in \binom{[r] \setminus [j]}{2}$. Observe also that for every $i \in [r] \setminus \{j\}$,

$$|\delta_G(\mathcal{V}^{(i)}, \mathcal{V}^{(j)})| = |\delta_G(\mathcal{V}^{(i)}, \partial_G(\mathcal{V}^{(j)}))|.$$

From Claim 3 and the fact that $\mathcal{V}^{*(i)} = \mathcal{V}^i$, for $i \in [r] \setminus \{j\}$, the last quantity is equal to $|\delta_{G_\mathcal{Q}}(\mathcal{V}^{*(i)}, \partial_G(\mathcal{V}^{*(j)}))| = |\delta_{G_\mathcal{Q}}(\mathcal{V}^{*(i)}, \mathcal{V}^{*(j)})| = \alpha(i, j)$.

Finally, as $\mathcal{V}^*$ is a solution of LA for $I = (G_\mathcal{Q}, r, \lambda|_Q, \alpha)$ and the image of λ contains j for all vertices in $V(G) \setminus Q$, $\mathcal{V}$ satisfies Condition 2. This finishes the proof that $\mathcal{V}$ is a solution of SHCLA for I.

In what follows, we assume that there exists a solution of SHCLA for (I, S). Fix some $j \in [r]$ such that there exists some solution $\mathcal{V}$ of SHCLA for (I, S) such that

A. $\mathcal{V}$ is $w \cdot f_1(w)$-bounded out of j and
B. $\partial_G(\mathcal{V}^{(j)}) \subseteq S \subseteq \mathcal{V}^{(j)}$.

From the second inclusion relation of **B**, $j \in \lambda(v)$ for all $v \in S$, therefore $j \in L$ and this choice of j will be considered during the application of **shcla-solver**(I, S) in Line 3. Let $\tilde{\mathcal{W}}^+$ be the set of subgraphs of G, produced by the algorithm in Line 10, given that j is chosen in Line 3.

Let $\tilde{\mathcal{W}}$ be the set containing all graphs in $\mathcal{C}(G \setminus S)$ that contain a vertex from a graph in $\tilde{\mathcal{W}}^+$.

Because of the first inclusion relation of **B**, the vertex set of every connected component in $\mathcal{C}(G \setminus S)$ should be either a subset of $\mathcal{V}^{(j)}$ or a subset of in $V(G) \setminus \mathcal{V}^{(j)}$. This distinction partitions $\mathcal{C}(G \setminus S)$ into sets $\mathcal{C}^{\mathcal{V}}_{\text{in}}$ and $\mathcal{C}^{\mathcal{V}}_{\text{out}}$ respectively. As $S \subseteq \mathcal{V}^{(j)}$, we obtain that

$$V(\bigcup \mathcal{C}_{\text{out}}) = V(G) \setminus \mathcal{V}^{(j)}. \tag{2}$$

Among all solutions of SHCLA for (I, S), satisfying **A** and **B** for this specific j, we choose $\mathcal{V}$ such that $|(\mathcal{W} \setminus \tilde{\mathcal{W}}) \cap \mathcal{C}^{\mathcal{V}}_{\text{out}}|$ is minimized. Our objective is to prove that **shcla-solver**(I, S) may output $\mathcal{V}$ as a solution for some choice of $\mathcal{Q}$ and $\mathcal{V}^*$. Before we determine which are these choices, we prove the following two claims.

Claim 4. $(\mathcal{W} \setminus \tilde{\mathcal{W}}) \cap \mathcal{C}^{\mathcal{V}}_{\text{out}} = \emptyset$.

Proof of the Claim: Suppose that there is a graph C in $\mathcal{W} \setminus \tilde{\mathcal{W}}$ that also belongs in $\mathcal{C}^{\mathcal{V}}_{\text{out}}$. We set $C^+ = G^+[V(C)]$. Let $\alpha_{C^+} : \binom{r}{2} \to [r]$ such that $\alpha_{C^+}(i,j) = |\delta_G(\mathcal{V}^{(i)} \cap V(C^+), \mathcal{V}^{(j)} \cap V(C^+))|$. We notice that C^+ belongs in $\mathbf{rep}(\mathcal{W}^+, \alpha_{C^+})$. The fact that $C \notin \tilde{\mathcal{W}}$ implies that $C^+ \notin \tilde{\mathcal{W}}^+$ and this means that $|\mathbf{rep}(\mathcal{W}^+, \alpha_{C^+})| \geq w$, therefore

$$|\tilde{\mathcal{W}}^+ \cap \mathbf{rep}(\mathcal{W}^+, \alpha_{C^+})| \geq w \tag{3}$$

We next prove that

$$|\mathcal{C}^{\mathcal{V}}_{\text{out}}| \leq w \tag{4}$$

To prove (4), observe first that, since G is connected and $\mathcal{C}^{\mathcal{V}}_{\text{out}}$ is a collection of connected components of $G \setminus S$, each member of $\mathcal{C}^{\mathcal{V}}_{\text{out}}$ contributes at least one unit to $|\delta_G(V(C), S)|$. Therefore $|\mathcal{C}^{\mathcal{V}}_{\text{out}}| \leq |\delta_G(V(\bigcup \mathcal{C}^{\mathcal{V}}_{\text{out}}), S)| \leq |\delta_G(V(G) \setminus \mathcal{V}^{(j)}, \mathcal{V}^{(j)})| = \sum_{i \in [r] \setminus \{j\}} \alpha(i,j) \leq w$ and (4) follows.

From (3) and (4), it follows that, among the graphs in $\tilde{\mathcal{W}}^+ \cap \mathbf{rep}(\mathcal{W}^+, \alpha_{C^+})$, there is at least one, say $C^{+\prime}$, that do not belong in $\mathcal{C}^{\mathcal{V}}_{\text{out}}$ and

$$|V(C^{+\prime} \setminus S)| \leq |V(C^+) \setminus S|. \tag{5}$$

As $C^{+\prime} \in \mathbf{rep}(\mathcal{W}^+, \alpha_{C^+})$, we have that $(C^{+\prime}, r, \lambda|_{V(C^{+\prime})}, \alpha_{C^+})$ is a YES-instance of CLA and let $\mathcal{V}_{C^{+\prime}}$ be a solution for this instance.

Let $\mathcal{V}_{C^+}$ be an r-allocation of $V(C^+)$ such that $\mathcal{V}^{(j)}_{C^+} = V(C^+)$ and $\mathcal{V}^{(i)}_{C^+} = \emptyset$ for $i \in [r] \setminus \{j\}$. We now set $\mathcal{V}' = \mathcal{V}'' \cup \mathcal{V}_{C^{+\prime}} \cup \mathcal{V}_{C^+}$, where $\mathcal{V}'' = \mathcal{V} \cap (V(G) \setminus (V(C^+) \cup V(C^{+\prime})))$. Observe that $\mathcal{V}'$ is a solution of CLA for I. Let $C' = G[C^{+\prime} \setminus S]$. Observe that (5) implies that $|V(C')| < |V(C)|$ and this yields Condition **A** for $\mathcal{V}'$. As both C and $C' = G[C^{+\prime} \setminus S]$ are connected components of $G \setminus S$, **B** holds for $\mathcal{V}'$ as well. We conclude that $\mathcal{V}'$ is a solution of SHCLA for (I, S). As $C' \in \tilde{\mathcal{W}}$ and $C \notin \tilde{\mathcal{W}}$ we have that $|(\mathcal{W} \setminus \tilde{\mathcal{W}}) \cap \mathcal{C}^{\mathcal{V}'}_{\text{out}}| < |(\mathcal{W} \setminus \tilde{\mathcal{W}}) \cap \mathcal{C}^{\mathcal{V}}_{\text{out}}|$, a contradiction to the choice of $\mathcal{V}$. The claim follows. ◇

Claim 5. *$\mathcal{Y} \subseteq \mathcal{C}^{\mathcal{V}}_{\text{out}}$ and $\mathcal{Z} \subseteq \mathcal{C}^{\mathcal{V}}_{\text{in}}$.*

Proof of the Claim: The fact that $\mathcal{Y} \subseteq \mathcal{C}^{\mathcal{V}}_{\text{out}}$ follows directly by the definition of $\mathcal{Y}$. To prove that $\mathcal{Z} \subseteq \mathcal{C}^{\mathcal{V}}_{\text{in}}$, consider some $C \in \mathcal{Z}$, which means that either $|V(C)| > w \cdot f_1(w)$ or $|N_G(V(C))| > w$. In the first case, $C \in \mathcal{C}^{\mathcal{V}}_{\text{in}}$, because of **A**. In the second case, assume towards a contradiction, that $\mathcal{C} \in \mathcal{C}^{\mathcal{V}}_{\text{out}}$. Notice that $\sum_{i\in[r]\setminus\{j\}} \alpha(i,j) \geq |\delta_G(C, \mathcal{V}^{(j)})|$ Also, $|\delta_G(C, \mathcal{V}^{(j)})| = |\delta_G(C, \partial_G(\mathcal{V}^{(j)}))|$. By the second inclusion relation of **B**, $|\delta_G(C, \partial_G(\mathcal{V}^{(j)}))| = |\delta_G(C, S)|$. Take also in mind that $|\delta_G(C, S)| \geq N_G(V(C))$. All together we have a contradiction because $w \geq \sum_{i\in[r]\setminus\{j\}} \alpha(i,j) \geq |\delta_G(C, \mathcal{V}^{(j)})| \geq |\delta_G(C, S)| \geq N_G(V(C)) > w$ which is a contradiction that completes the proof of the claim. ◇

Let $\mathcal{C}^+_{\text{out}} = \{G^+[V(C)] \mid C \in \mathcal{C}_{\text{out}}\}$. From above two claims we obtain that $\mathcal{Y} \subseteq \mathcal{C}_{\text{out}} \subseteq \mathcal{Y} \cup \tilde{\mathcal{W}}$. This, together with (4), imply that the set $\mathcal{C}^+_{\text{out}}$ will be one of the choices for the set $\mathcal{Q}$ in Line 11 of **shcla-solver**(I, S). Notice that $|Q| = |V(\bigcup \mathcal{C}_{\text{out}})| + |N_G(V(\bigcup \mathcal{C}_{\text{out}}))|$ which, because of (2), is equal to $|V(G) \setminus \mathcal{V}^{(j)}| + |\partial_G(\mathcal{V}^{(j)})|$. As **A** holds for $\mathcal{V}$, it follows that $|V(G) \setminus \mathcal{V}^{(j)}| \leq w \cdot f_1(w)$. Moreover, $|\partial_G(\mathcal{V}^{(j)})| \leq |\delta_G(V(\bigcup \mathcal{C}^{\mathcal{V}}_{\text{out}}), S)| \leq |\delta_G(V(G)\setminus\mathcal{V}^{(j)}, \mathcal{V}^{(j)})| = \sum_{i\in[r]\setminus\{j\}} \alpha(i,j) \leq w$. We conclude that $|Q| \leq w \cdot (f_1(w) + 1)$. Because of this, the choice $\mathcal{Q} := \mathcal{C}^+_{\text{out}}$ in Line 3 will make the algorithm pass the first test of Line 13.

Let now $\mathcal{V}^* = \mathcal{V} \cap Q$ and observe that this intersection does not remove any vertex from the parts of $\mathcal{V}$ that are different than $\mathcal{V}^{(j)}$ while from $\mathcal{V}^{(j)}$ it removes vertices that belong either to S or to the graphs of $\mathcal{C}(G \setminus S)$ that do not intersect Q. Using the fact that $\mathcal{V}$ is a solution for (I, S) and the definition of $\lambda_{j,S}$, one can easily observe that $\mathcal{V}^* \in \mathbf{sol}(G_{\mathcal{Q}}, r, \lambda_{j,S}|_Q, \alpha)$. Therefore, the algorithm may choose this solution $\mathcal{V}^*$ in the second test of Line 13, extend it to $\mathcal{V}$ in Lines 15 and 16, and output $\mathcal{V}$ as a solution for (I, S) as required. This finish the proof of the correctness of the algorithm.

For the running time observe first that $G_{\mathcal{Q}}$ is the union of at most w connected graphs, therefore it has at most w connected components. Recall also that $f_1(w) = 2^{O(w\cdot\log w)}$. From Lemma 1, the set $\mathbf{sol}(G_{\mathcal{Q}}, r, \lambda_{j,S}|_Q, \alpha)$ can be constructed in $|Q|^{O(w)} \cdot 2^{O(w\cdot\log r)} = (2^{O(w\cdot\log w)})^{O(w)} \cdot 2^{O(w\cdot\log r)} = 2^{O(w^2\cdot\log w)}$ steps. Notice now that the choices of Line 11 are possible only when $|\mathcal{Y}^+| \leq w$. Moreover, from Observation 4, $|\tilde{\mathcal{W}}^+| \leq 2^{O(w)}$. In total, $|\mathcal{Y}^+| + |\tilde{\mathcal{W}}^+| = 2^{O(w)}$, thus there are $2^{O(w^2)}$ choices for the set $\mathcal{Q}$ in Line 11. As the classification of the connected components of $G \setminus S$ in to the sets $\mathcal{W}$, $\mathcal{Y}$, and $\mathcal{Z}$ can be done in $O(w \cdot n)$ steps, the claimed running time follows. □

B Applications

In this appendix we present the applications of our FPT-algorihtm fir the LA problem. Each subsection is devoted to a particular problem and contains all the necessary definitions and TFPT reductions to the List Allocation problem.

B.1 A parameterization of Min-Sum-Multiway Cut

The Min-Sum-Multiway Cut problem is formally defined as follows:

Min-Sum-Multiway Cut
Input: An undirected graph G, $w, r \in \mathbb{Z}_{\geq 0}$, and a set $T \subseteq V(G)$, with $|T| = r$.
Parameter: $w \cdot r$.
Output: A partition $\{\mathcal{P}_1, \ldots, \mathcal{P}_r\}$ of $V(G)$ such that for every $i \in [r]$, it holds that $|\mathcal{P}_i \cap T| = 1$ and $|\delta_G(\mathcal{P}_i, V(G) \setminus \mathcal{P}_i)| \leq w$, or a correct report that no such partition exists.

Theorem 2. *There exists an algorithm that, given as input an instance (G, T, w, r) of* Min-Sum-Multiway Cut*, solves this problem in $2^{O((wr)^2 \log wr)} \cdot n^4 \cdot \log n$ steps.*

Proof: Given a quadruple $I = (G, T, w, r)$, we fix a bijection $\mu : V(T) \to [r]$ and we define $\lambda : V(G) \to 2^{[r]}$ such that

$$\lambda(x) = \begin{cases} [r] & \text{if } x \in V(G) \setminus T \\ \{\mu(x)\} & \text{if } x \in T \end{cases}$$

We now define a family $\mathcal{U}(I)$ of instances of LA containing every (G, r, λ, α) for which $\alpha : \binom{[r \cdot w]}{2} \to \mathbb{Z}_{\geq 0}$ such that $\forall i \in [r], \sum_{j \in [r] \setminus i} |\delta_G(\alpha(i), \alpha(j))| \leq w$. Notice that I is a YES-instance of Min-Sum-Multiway Cut if there exists some $I' \in \mathcal{U}(I)$ that is a YES-instance of LA. This is a TFPT-reduction to the LA problem as $|\mathcal{U}(I)| = 2^{O(rw)}$ and the result follows by Theorem 1. □

B.2 Edge Cutting into Many Components

Consider the following problem:

Edge cutting into many Components
Input: A tuple (G, w, r) where G is an undirected graph and $w, r \in \mathbb{Z}_{\geq 0}$.
Parameter: w.
Output: A set $F \subseteq E(G)$ such that $|F| \leq w$ and $G \setminus F$ contains at least r connected components, or a correct report that such a set does not exist.

Theorem 3. *There exists an algorithm that, given an input (G, w, r) of the* Edge cutting into many Components *problem, solves the problem in $2^{O(w^2 \log w)} \cdot n^4 \cdot \log n$ steps.*

PROOF: In the introduction we explained, by a straightforward TFPT-reduction to LIST ALLOCATION that the above problem can be solved in $2^{O(w^2 \log w)} \cdot n^4 \cdot \log n$ steps in the case where G is a connected graph. For the general case, assume that $|\mathcal{C}(G)| = m$ and run this algorithm for each $C_i \in \mathcal{C}(G)$, for $w' \in [w]$ and $r' \in [w+1]$. This permits us, for every $C \in \mathcal{C}(G)$, to set up a function $f_C : \{0, \ldots, w\} \to [w+1]$ such that $f_C(x)$ is the maximum number of connected components that may appear when we remove from G_i a set of at most x edges. We say that two connected components C and C' of G are *equivalent* if $f_C = f_{C'}$ and this equivalence relation partitions $\mathcal{C}(G)$ into $2^{O(w \cdot \log w)}$ equivalent classes. As no more than w graphs may intersect a certificate of a solution of the problem, an equivalent instance (G', w, r') is created if G' is the graph occuring from G after removing from each equivalent class with more than w elements all by w of them. Also $r' = r - m^*$ where m^* is the number of connected components that where removed. Observe now that $m' = |\mathcal{C}(G')| = 2^{O(w \cdot \log w)}$ and that if $r' > m' + w$, then (G', w, r') is a NO-instance. From now on we can assume that $r' = 2^{O(w \cdot \log w)}$.

Let now $\mathcal{T}$ be the set containing all possible tuples $(w_1, \ldots, w_{m'})$ such that $\sum_{i \in [r']} w_i \leq w$ and assume that $\mathcal{C}(G') = \{C'_1, \ldots, C'_{m'}\}$. It is easy to see that (G', w, r') is a YES-instance if an only if there exists some $(w_1, \ldots, w_{m'})$ in $\mathcal{T}$ such that $\sum_{i \in [m']} f_{C_i}(w_i) \geq r'$. The claimed running time follows as $|\mathcal{T}| = 2^{O(w^2 \cdot \log w)}$. □

B.3 Bounded List Allocation

In this section we study the parameterized complexity of the following enhancement of the LIST ALLOCATION problem. As we mentioned in the introduction, this problem is useful for the algorithm that we describe for the CUTTING A SPECIFIC NUMBER OF VERTICES problem. Also, this problem will also be useful for the TFPT-reductions of Subsection B.5.

BOUNDED LIST ALLOCATION (BLA)
Input: A tuple $I = (G, r, q, \lambda, \alpha, \beta)$ where G is a graph, $r, q \in \mathbb{Z}_{\geq 1}$, $\lambda : V(G) \to 2^{[r+q]}$, $\alpha : \binom{[r+q]}{2} \to \mathbb{Z}_{\geq 0}$, and $\beta : [q] \to \mathbb{N}_{\geq 0}$.
Parameter: $w = \sum \alpha + \sum \beta$.
Output: An r-allocation $\mathcal{V}$ of $V(G)$ such that

1. $\forall \{i, j\} \in \binom{[r+q]}{2}$, $|\delta_G(\mathcal{V}^{(i)}, \mathcal{V}^{(j)})| = \alpha(i, j)$,
2. $\forall v \in V(G), \forall i \in [r]$, if $v \in \mathcal{V}^{(i)}$ then $i \in \lambda(v)$, and
3. $|\mathcal{V}^{(r)}| = \beta(i)$,

or a correct report that such an r-allocation does not exist.

We need the following result.

Proposition 3 (Kirousis *et al.* [22]). *For every $w \in \mathbb{Z}_{>0}$, every graph G where $|E(G)| \geq w \cdot (|V(G)| - 1)$ contains a w-connected subgraph.*

Given a graph H and a $w \in \mathbb{Z}_{>0}$, we say that a subgraph H of G is a *w-edge-connected core of G* if every connected component of H is w-edge-connected and H has a maximal number of edges.

Lemma 10. *For every $w \in \mathbb{Z}_{>0}$, every graph G where $|E(G)| \geq w \cdot (|V(G)| - 1)$ contains a unique w-edge-connected core that can be found in $O(n^4)$ steps.*

Proof: The claimed w-edge-connected core exists because of Proposition 3. The algorithm that finds it repetitively removes from G edges of min-cuts of size at most $w-1$ (each can be found in $O(n^3)$ steps according to [28]) until this is not possible anymore (isolated vertices, when appearing during this procedure, are removed). □

Lemma 11. *There is an $O(n^4)$-step algorithm that given in instance $I = (G, r, q, \lambda, \alpha, \beta)$ of* BLA*, outputs an equivalent instance $I' = (G', r, q, \lambda', \alpha, \beta)$.*

Proof: Let H be the $(w+1)$-edge-connected core of G, that can be computed in $O(n^4)$ steps, according to the procedure of Lemma 10. For each $C \in \mathcal{C}(H)$ such that $|V(C)| \geq w+1$, compute $L_C = \cap_{x \in V(C)} \lambda(x)$ identify in G all vertices of $V(C)$ to a single vertex v_C. We denote by G^* the resulting graph. Also we construct the graph G' by introducing, for each $C \in \mathcal{C}(H)$, a new edge e_C with multiplicity $w+1$ in G^*, and and identifying v_C with one of the endpoints of e_C. Finally we define $\lambda' : V(G') \to 2^{[r+q]}$ such that

$$\lambda' = \lambda|_{V(G) \setminus V(\bigcup \mathcal{C}(H))} \cup \bigcup_{C \in \mathcal{C}(H)} \{(x, L_C) \mid x \in V(e_C)\}.$$

Using the properties of edge connectivity, it can be proved that the resulting graph G^* does not contain any $(w+1)$-edge-connected subgraph, therefore, from Proposition 3 G' has $O(w \cdot n(G'))$ edges. It remains to verify that $I = (G', r, q, \lambda, \alpha, \beta)$ is an equivalent intense.

Suppose that $\mathcal{V}$ is a solution of BLA for $I = (G, r, q, \lambda, \alpha, \beta)$. Observe first that all the connected components of H that have more than w vertices are allocated into the first r (unbounded) boxes. Moreover, all the vertices of a connected component of H should belong in the same part of $\mathcal{V}$, otherwise the removal of at most w edges would be able to separate different parts of $\mathcal{V}$. These two facts imply that $I' = (G', r, q, \lambda', \alpha, \beta)$ is a YES-instance of BLA.

Assume now that $\mathcal{V}'$ is a solution of BLA for $I' = (G', r, q, \lambda', \alpha, \beta)$. As the endpoints of each e_C should go altogether to one of the first r parts of a $\mathcal{V}'$ we can enhance $\mathcal{V}$ to a solution solution $\mathcal{V}$ for I by assigning each connected component C in $\mathcal{C}(H)$ of more than w vertices to the same box as the one containing e_C. □

Given a set $R \subseteq \mathbb{Z}_{\geq 0}$ and an integer $x \in \mathbb{Z}_{\geq 0}$. we denote by $R \oplus x$ then set ocuring from R if we add x to all the elements of R.

Lemma 12. *If there exists an algorithm that solves* BLA *that runs in $2^{O(w \cdot \log w)} \cdot n^4 \cdot \log n$ steps.*

Proof: We give an TFPT-reduction of BLA to LA.

We assume that $\forall i \in [q]\ \beta(i) > 0$, otherwise this instance is equivalent to some instance that has a smaller q and where this condition holds.

Given an instance $I = (G, r, q, \lambda, \alpha, \beta)$ of BLA, we build the instance $I' = (G', r', \lambda', \alpha')$ of LA as follows: We first define G_s to be the graph obtained by subdividing once each edge of G. In G_s we denote by V_s the subdivision vertices and we denote by $\tau : E(G) \to V_s$ the bijection mapping each edge of $E(G)$ to the vertex of V_s that subdivided it. The graph G' is obtained by adding in G_s a set V_p of new vertices and making each one of them adjacent to exactly one of the original vertices of G. We denote by $\rho : V(G) \to V_p$ the bijection mapping each $v \in V(G)$ to its unique neighbor in V_s. This completes the construction of G'.

We set $r' = (r+q) + q + \binom{r+q}{2}$, $P = \{r+1, \ldots, r+q\}$, $P^+ = P \oplus r$, $S = \{(r+q)+q+1, \ldots, (r+q)+q+\binom{r+q}{2}\}$ we fix a bijection $\mu : \binom{r+q}{2} \to S$, and we define $\lambda' : V(G') \to [r']$ such that

$$\lambda'(x) = \begin{cases} \lambda(x) & \text{if } x \in V(G) \\ \{\mu(a,c) \mid (a,c) \in \lambda(y) \times \lambda(z)\} & \text{if } x \in V_s \\ & \quad \text{(we set } \{y,z\} = \tau^{-1}(x)) \\ \lambda(\rho^{-1}(x)) \setminus P \cup ((\lambda(\rho^{-1}(x)) \cap P) \oplus r) & \text{if } x \in V_p \end{cases}$$

Also, we define $\alpha' : \binom{[r']}{2} \to \mathbb{Z}_{\geq 0}$ such that

$$\alpha'(x,y) = \begin{cases} b(i) & \text{if } \{x,y\} = \{i, i+r\} \text{ for some } i \in P \\ \alpha(\mu^{-1}(y)) & \text{if } x \in [r+q] \text{ and } y \in S \\ 0 & \text{otherwise} \end{cases}$$

Let $\mathcal{V}$ be a solution for I. We define a solution $\mathcal{V}'$ for I' such that

- $\forall_{i \in [r]}\ \mathcal{V}'^{(i)} = \mathcal{V}^{(i)} \cup p(\mathcal{V}^{(i)})$,
- $\forall_{i \in P}\ \mathcal{V}'^{(i)} = \mathcal{V}^{(i)}$,
- $\forall_{i \in P^+}\ \mathcal{V}'^{(i)} = \rho(\mathcal{V}^{(i)})$, and
- $\forall_{i \in S}\ \mathcal{V}'^{(i)} = \{x \in V_s \mid \text{if } \{a,c\} = \mu^{-1}(i) \text{ and } \{y,z\} = \tau^{-1}(x) \text{ then } \mathcal{V}^{(a)} \text{ contains one of the two vertices in } \{y,z\} \text{ and } \mathcal{V}^{(c)} \text{ contains the other}\}$,

and observe that $\mathcal{V}'$ is a solution for I'.

Let now $\mathcal{V}'$ be a solution for I'. The definition of λ' implies the following

$$V(G) \subseteq \bigcup_{i \in [r+q]} \mathcal{V}'^{(i)}, \tag{6}$$

$$V_s = \bigcup_{i \in S} \mathcal{V}'^{(i)}, \tag{7}$$

$$\bigcup_{i \in P^+} \mathcal{V}'^{(i)} \subseteq V_p. \tag{8}$$

Out first step is to prove that $\forall_{i \in P}\ |\mathcal{V}'^{(i)}| = \beta(i)$. For this let $i \in P$. Notice that the set $F_i = \delta_{G'}(\mathcal{V}'^{(i)}, \mathcal{V}'^{(i+r)})$ contains $\alpha'(i, i+r) = \beta(i)$ edges, each with

one endpoint in $\mathcal{V}'^{(i)}$ and one endpoint in $\mathcal{V}'^{(i+r)}$. From (8) and the fact that vertices in V_p have degree 1 in G' we obtain that $|\mathcal{V}'^{(i+r)}| \geq b$. We also claim that there is no other vertex of V_p in $\mathcal{V}'^{(i+r)}$. Indeed, if x is such a vertex, then $\rho^{-1}(x)$ should belong in $\mathcal{V}'^{(i)}$ because i is the only index in P for which $\alpha'(i, i+r) > 0$, a contradiction to the fact that $|F| = b$. We just proved that

$$|\mathcal{V}'^{(i+r)}| = \beta(i). \tag{9}$$

Using again the fact that i is the only index in $[r+q]$ for which $\alpha'(i, i+r) > 0$, we have that $\rho^{-1}(\mathcal{V}'^{(i+r)}) \subseteq \mathcal{V}'^{(i)}$. In order to prove the equality, assume to the contrary that $\mathcal{V}'^{(i)}$ contains a vertex z of G' where $z \notin \rho^{-1}(\mathcal{V}'^{(i+r)})$. By (6), $z \in \mathcal{V}'^{(i)}$ implies that $z \in V(G)$. Let j be the index in $[r']$ for which $\rho(z) \in \mathcal{V}'^{(j)}$. From the fact that $\alpha(i, j) > 0$ and (7) we obtain that $j \in P^+$. By the definition of λ', $j \in P^+$ and the the fact that $\alpha(i, j) > 0$ imply that $j = i + r$. This means that $\rho(z) \in \mathcal{V}'^{(i+r)}$, which implies that $z \in \rho^{-1}(\mathcal{V}'^{(r+1)})$, a contradiction. This finish the proof that $\rho^{-1}(\mathcal{V}'^{(i+r)}) = \mathcal{V}'^{(i)}$. As $|\rho^{-1}(\mathcal{V}'^{(i+r)})| = |\mathcal{V}'^{(i+r)}|$, from (9) we conclude that $|\mathcal{V}'^{(i)}| = \beta(i)$.

Ir remains now to consider the $(r+q)$-allocation $\mathcal{V} = \{\mathcal{V}'^{(1)}, \ldots, \mathcal{V}'^{(r+q)}\} \cap V(G)$. Clearly $\mathcal{V}$ is an r-allocation of $V(G)$ where, from (6), $|\mathcal{V}^{(i)}| = \beta(i)$, for every $i \in P$. Using the second line of the definition of α', it is easy to verify that $\mathcal{V}$ is a solution of BLA for the instance I.

Notice now that $|V(G')| = |V(G)| + |E(G)|$ and $w' = \sum \alpha' = 2 \cdot \sum \alpha + \sum \beta = O(w)$. Also, because of Lemma 11, we can assume that $|E(G)| = O(w \cdot |V(G)|)$. As $|V(G')| = O(w \cdot |V(G)|)$, the result now follows applying the algorithm of Theorem 1 on I'. □

B.4 A parameterization of List Digraph Homomorphism

Let G and H be directed graphs where G is simple and H may have loops but not multiple directed edges. Let also $\lambda : V(G) \to 2^{V(H)}$. We denote by $E_1(H)$ the loops of H and by $E_2(H)$ the edges of H between distinct vertices. An *λ-list H-homomorphism* of G is a function $\chi : V(G) \to V(H)$ such that for every $v \in V(G)$, $\chi(v) \in \lambda(v)$, and such that for every $u, v \in V(G)$, it holds that $(u, v) \in E(G) \to (\chi(u), \chi(v)) \in E(H)$. Given a list H-homomorphism χ of G and an edge $e = \{a, b\} \in E_2(H)$ we define the χ-arc charge of e as the set $C(e) = \{(u, v) \in E(G) \mid \chi(u) = a \text{ and } \chi(v) = b\}$. Given a function $\alpha : E_2(H) \to \mathbb{Z}_{\geq 0}$, we say that χ is *α-arc-specified* if $|C(e)| = \alpha(e)$, for every $e \in E_2(H)$.

Arc-Specified List Digraph Homomorphism (ASLDH)
Input: A tuple (G, H, λ, α) where G is a digraph, H is a digraph with possible loops and without multiple directed edges, α is a function from $E_2(H)$ to $\mathbb{Z}_{\geq 0}$, and λ is a function from $V(G)$ to $2^{V(H)}$.
Parameter: $w = \sum \alpha$.
Output: An α-arc-specified λ-list H-homomorphism of G, or a correct report that no such λ-list H-homomorphism exists.

Lemma 13. *There is an $O(n^4)$-step algorithm that given in instance $I = (G, H, \lambda, \alpha)$ of* ASLDH*, outputs an equivalent instance $I' = (G', H, \lambda', \alpha)$ of the same problem where $|E(G')| = O(w \cdot |V(G')|)$.*

Proof: Let $\tilde{G}$ be the underlying graph of G (multiplicities of edges of opposite direction are summed up) and let H be the $(w+1)$-edge-connected core of $\tilde{G}$. For each connected component C of H we identify in G all vertices of C to a single vertex v_C and we update λ to λ' so that if $x \notin \{v_C \mid C \in \mathcal{H}(C)\}$, then $\lambda'(x) = \lambda(v)$ and if $x = v_C$, then $\lambda'(x) = \cup_{y \in V(C)} \lambda(y)$. Following the same arguments as in the proof of Lemma 11, one can see that $I' = (G', H, \lambda', \alpha)$ is an equivalent instance where $|E(G')| = O(w \cdot |V(G')|)$. □

Theorem 4. *There exists an algorithm that, given as input an instance (G, H, λ, α) of* Arc-Specified List Digraph Homomorphism*, returns and answer to this problem in $2^{O(w^2 \cdot \log w)} \cdot n^4 \cdot \log n$ steps.*

Proof: Given an instance $I = (G, H, \lambda, \alpha)$ of Arc-Specified List Digraph Homomorphism we generate an instance $I' = (G', r, \lambda', \alpha')$ of LA, as follows:

- $G' = (V', E')$, where
  - $V' = V \cup V_F \cup V_L$, where $V = V(G)$, $V_F = \{f_{uv} \mid (u,v) \in E(G)\}$, and $V_L = \{\ell_{uv} \mid (u,v) \in E(G)\}$ and
  - $E' = E \cup E_F \cup E_L$, where $E = \{\{f_{uv}, \ell_{uv}\} \mid (u,v) \in E(G)\}$, $E_F = \{\{u, f_{uv}\} \mid (u,v) \in E(G)\}$, $E_L = \{\{\ell_{uv}, v\} \mid (u,v) \in E(G)\}$.
- $r = |V(H)| + 2 \cdot |E_2(H)|$ and let $\sigma : V(\tilde{H}) \to [r]$ be some bijection where $\tilde{H}$ is the graph obtained from H by subdividing twice each of its arcs that are not loops. For each arc $(x,y) \in E_2(H)$, we denote its corresponding path in $\tilde{H}$ as P_{xy}, where $V(P_{xy}) = \{x, \tilde{f}_{xy}, \tilde{\ell}_{xy}, y\}$.
- $\lambda' : V(G') \to [r]$ such that

$$\lambda'(w) = \begin{cases} \{\sigma(x) \mid x \in \lambda(w)\} & \text{if } w \in V \\ \{\sigma(\tilde{f}_{xy}) \mid x \in \lambda(u) \wedge y \in \lambda(v) \wedge x \neq y\} \cup \\ \{\sigma(x) \mid x \in \lambda(u) \cap \lambda(v) \wedge (x,x) \in E_1(H)\} & \text{if } w = f_{uv} \in V_F \\ \{\sigma(\tilde{\ell}_{xy}) \mid x \in \lambda(u) \wedge y \in \lambda(v) \wedge x \neq y\} \cup \\ \{\sigma(x) \mid x \in \lambda(u) \cap \lambda(v) \wedge (x,x) \in E_1(H)\} & \text{if } w = \ell_{uv} \in V_L. \end{cases}$$

- $w = \max \alpha$.
- $\alpha' : \binom{[r]}{2} \to \mathbb{Z}_{\geq 0}$ such that

$$\alpha'(i,j) = \begin{cases} \alpha(x,y) & \text{if there exists some } (x,y) \in E_2(H) \text{ such that} \\ & \quad (i,j) \in \big\{(\sigma(x), \sigma(\tilde{f}_{xy})), (\sigma(\tilde{f}_{xy})), \sigma(\tilde{\ell}_{xy})), (\sigma(\tilde{\ell}_{xy}), \sigma(y))\big\} \\ 0 & \text{otherwise.} \end{cases}$$

Let $\chi : V(G) \to V(H)$ be an α-arc-specified λ-list H-homomorphism of G. We construct an r-allocation $\mathcal{V}$ of $V(G')$ as follows:

- for every $u \in V = V(G)$, u belongs to the part $\mathcal{V}^i$, where $i = \sigma(\chi(u))$
- for every $f_{uv} \in V_F$, f_{uv} belongs to the part $\mathcal{V}^i$, where

$$i = \begin{cases} \sigma(\chi(u)) \text{ if } \chi(u) = \chi(v) \\ \sigma(\tilde{f}_{xy}) \quad \text{if } x = \chi(u) \neq y = \chi(v) \end{cases}$$

- for every $\ell_{uv} \in V_L$, ℓ_{uv} belongs to the part $\mathcal{V}^i$, where

$$i = \begin{cases} \sigma(\chi(u)) \text{ if } \chi(u) = \chi(v) \\ \sigma(\tilde{\ell}_{xy}) \quad \text{if } x = \chi(u) \neq y = \chi(v) \end{cases}$$

It is straightforward to verify that $\mathcal{V}$ is a solution for I'.

Now consider a solution $\mathcal{V}$ for I'. From $\mathcal{V}$, we define a mapping $\chi : V(G) \to V(H)$ so that for every $u \in V$, we have $\chi(u) = \sigma^{-1}(i)$ if and only if $u \in \mathcal{V}^{(i)}$. We claim that χ is an α-arc-specified λ-list H-homomorphism of G. For this, we investigate χ upon two conditions: firstly, we verify that χ is a λ-list H-homomorphism, and secondly that χ is α-arc-specified.

Let us prove that χ is a λ-list H-homomorphism. To see that $\chi(u) \in \lambda(u)$ for every $u \in V(G)$, let u be in the i-th part of $\mathcal{V}$. Since $i \in \lambda'(u)$, the construction of λ' implies that $\sigma^{-1}(i) \in \lambda(u)$, and thus $\chi(u) \in \lambda(u)$. To see that χ is an H-homomorphism, for an arbitrary edge $(u, v) \in E(G)$ we shall show that $(\chi(u), \chi(v)) \in E_1(H) \cup E_2(H)$. Let u and v respectively belong to $\sigma(x)$-th and $\sigma(y)$-th parts of $\mathcal{V}$, for some $x, y \in V(\tilde{H})$. Note that $x \in \lambda(u) \subseteq V(H)$ and $y \in \lambda(v) \subseteq V(H)$. There are two possibilities: $x \neq y$ or $x = y$.

Case 1: $x \neq y$. Since σ is a bijection, this means $\sigma(x) \neq \sigma(y)$. From the way we construct α', the vertices f_{uv} and ℓ_{uv} can be only allocated into the $\sigma(\tilde{f}_{xy})$-th part and the $\sigma(\tilde{\ell}_{xy})$-th part, respectively, in the solution $\mathcal{V}$. Furthermore, the construction of α' also implies $(x, y) \in E_2(H)$.

Case 2: $x = y$. This means $\sigma(x) = \sigma(y)$. The construction of α' implies f_{uv} and ℓ_{uv} are allocated into the $\sigma(x)$-th part of $\mathcal{V}$ as well. This, in turn, means that $\sigma(x) \in \lambda'(f_{uv})$ and $\sigma(x) \in \lambda'(\ell_{uv})$. Recall that $\lambda'(f_{uv})$ contains $\sigma(x)$ only when $(x, x) \in E_1(H)$. Hence, $(x, y) \in E_1(H)$.

Now we verify that χ is α-arc-specified. Consider an arc $e = (x, y) \in E_2(H)$. Note that for every directed edge (u, v) in the χ-arc charge $C(e)$, the (u, f_{uv}) of $E(G')$ contributes to $\alpha'(\sigma(x), \sigma(\tilde{f}_{xy}))$ exactly by one unit. Conversely, for every edge (u, f_{uv}) of $E(G')$ which contributes to $\alpha'(\sigma(x), \sigma(\tilde{f}_{xy}))$, we have $\chi(v) = y$ and thus the directed arc (u, v) contributes to $C(e)$ by one unit. This establishes that χ is α-arc-specified.

From Lemma 13, we can assume that $|E(G)| = O(w \cdot |V(G)|)$. The claimed running time follows from Theorem 1, given that $\sum \alpha' = 3 \cdot \sum \alpha = 3 \cdot w$ and that $|V(G')| = O(|E(G)|) = O(w \cdot |V(G)|)$. □

B.5 An FPT 2-approximation for tree-cut width

B.5.1. A special partitioning problem. Consider the following problem:

Special Partitioning
Input: An undirected graph G, a $w \in \mathbb{Z}_{\geq 0}$, a set $B \subseteq V(G)$, and a weight function $\gamma : B \to [2w]$ such that $\sum \gamma \leq 2w$.
Parameter: w.
Output: An $(r+1)$-allocation $\mathcal{V}$ of $V(G)$, for some $r \in \mathbb{Z}_{\geq 0}$, such that

1. there are at least two non-empty sets in $\mathcal{V}$,
2. $|\mathcal{V}^{(r+1)}| + r \leq w$,
3. $|\gamma(B \cap \mathcal{V}^{(i)})| \leq w$ for every $i \in [r]$, and
4. $|\delta_G(\mathcal{V}^{(i)}, V(G) \setminus \mathcal{V}^{(i)})| \leq w$ for every $i \in [r]$,

or a correct report that no such $(r+1)$-allocation exists.

Given a solution $\mathcal{V}$ for some instance of Special Partitioning, we call $\mathcal{V}^{(r)}$ the *central part* of $\mathcal{V}$, where $r = |\mathcal{V}|$.

Lemma 14. *Let $w \geq 2$, $\rho \geq 2$ and let $A = \{a_1, \ldots, a_\rho\}$ be a set of non-negative integers such that $\sum_{i \in [\rho]} a_i \leq 2w$ and $\forall_{i \in [\rho]}\ a_i \leq w$. Then there exists a partition $\mathcal{A}$ of A such that $2 \leq |\mathcal{A}| \leq w$ and such that for each $P \in \mathcal{A}$ it holds that $\sum_{x \in P} x \leq w$. Moreover, such a partition can be obtained in time $O(n \log n)$.*

Proof: Any element of A equal to zero can be added to an arbitrary part P of a given partition without increasing the sum of the elements in P. Hence, without loss of generality we assume that all elements of A are positive. When $w = 2$, a simple case analysis shows that there exists a partition as claimed in the statement. Therefore we consider the case when $w \geq 3$.

First, we order the elements of A in non-decreasing order, therefore we may assume that $a_1 \leq \cdots \leq a_\rho$. This can be done in the claimed running time.

Case 1. $a_{\rho-1} + a_\rho > w$. We create a partition $\mathcal{A}$ into three parts: the first part contains a_ρ, the second part contains $a_{\rho-1}$, and the last one one contains the remaining elements of A. Observe that for each $P \in \mathcal{A}$, $\sum_{x \in P} x \leq w$. In particular, this holds for the third part consisting of $A \setminus \{a_\rho, a_{\rho-1}\}$, due to the fact $\sum_{i \in [\rho]} a_i \leq 2w$ and $a_{\rho-1} + a_\rho > w$.

Case 2. $a_{\rho-1} + a_\rho \leq w$. We form an arbitrary partition $\mathcal{A}$ such that $|\mathcal{A}| = \lceil \frac{\rho}{2} \rceil$ and $|P| \leq 2$ for every $P \in \mathcal{A}$. From $\sum_{i \in [\rho]} a_i \leq 2w$ and the fact that $a_i \geq 1$, for every $i \in [\rho]$, it follows that $|\mathcal{A}| = \lceil \frac{\rho}{2} \rceil \leq \lceil \frac{2w}{2} \rceil \leq w$. Moreover, $a_i + a_j \leq a_{\rho-1} + a_\rho \leq w$ for any $\{i, j\} \in \binom{[\rho]}{2}$ and thus $\sum_{x \in P} x \leq w$ for every $P \in \mathcal{A}$. □

Lemma 15. *There exists an algorithm for* Special Partitioning *that runs in $2^{O(w^4 \cdot \log w)} \cdot n^4 \cdot \log n$ steps.*

PROOF: Let $I = (G, w, B, w)$ be an instance of SPECIAL PARTITIONING. We assume that $w \geq 2$ as, otherwise I is a NO-instance. In case $|\mathcal{C}(G)| \geq 2$ and $w(V(C) \cap B) \leq w$ for all $C \in \mathcal{C}(G)$, then let $\{C_1, \ldots, C_\rho\} = \mathcal{C}(G)$ and $a_i = w(B \cap C_i), i \in [\rho]$. By applying Lemma 14 for $A = \{a_1, \ldots, a_\rho\}$ we obtain a partition $\mathcal{A} = \{P_1, \ldots, P_\sigma\}$ of A. We then construct an allocation $\mathcal{V}$ of $V(G)$ such that $|\mathcal{V}| = \sigma + 1$, where for $i \in [\sigma]$, $\mathcal{V}^{(i)}$ contains all the vertices of the members of $\mathcal{C}(G)$ whose indices belong in P_i and $\mathcal{V}^{(\sigma+1)} = \emptyset$. Observe that $\mathcal{V}$ is a solution for I.

If $|\mathcal{C}(G)| = 1$ or if $w(V(C) \cap B) > w$ for some $C \in \mathcal{C}(G)$, then we generate the collection $\mathcal{U}(I)$ containing every instance $(G, r, 1, \lambda, \alpha, \beta)$ of BLA satisfying the following:

- $r \in [w]$,
- $\lambda : V(G) \to 2^{[r+1]}$ such that

$$\lambda(x) = \begin{cases} [r+1] & \text{if } x \in V(G) \setminus B \\ \{\mu(x)\} & \text{if } x \in B \end{cases}$$

  with $\mu : B \to [r+1]$ such that for every $i \in [r]$, $w(\mu^{-1}(i)) \leq w$,
- $\alpha : \binom{[r+1]}{2} \to \mathbb{Z}_{\geq 0}$ such that

$$\sum \alpha > 0 \quad \text{and for every} \quad i \in [r], \sum_{j \in [r] \setminus \{i\}} \alpha(i,j) \leq w,$$

- $\beta(1) \leq w - r$.

In the above construction, keep in mind that $\sum \alpha = O(w^2)$. It is easy to verify that I is a YES-instance of SPECIAL PARTITIONING if and only if there is some $I' \in \mathcal{U}(I)$ such that I' is a YES-instance of BLA. Notice also that $|\mathcal{U}(I)| = (w+1) \cdot 2^{O(w \cdot \log r)} \cdot 2^{O(w^2 \cdot \log w)} \cdot O(w)$. The claimed running time follows combining Theorem 1 and Lemma 12. □

B.5.2. The FPT 2-approximation

Tree-cut width. A family of possibly empty subsets $X_1, \ldots, X_w$ of a finite set X is a *near-partition* of X if they are pairwise disjoint and $\bigcup_{i=1}^{w} X_i = X$. A *tree-cut decomposition* of G is a pair $(T, \mathcal{X})$ which consists of a tree T and a near-partition $\mathcal{X} = \{X_t \subseteq V(G) : t \in V(T)\}$ of $V(G)$. A set in the family $\mathcal{X}$ is called the *bag* of the tree-cut decomposition. From now on we refer to the vertices of T as *nodes*. A node $v \in T$ is *trivial* in $(T, \mathcal{X})$ if $X_t = \emptyset$.

For every edge $e = \{u, v\}$ of T, let T_u and T_v be the two components in $T \setminus e$ which contain u and v respectively. Note that $\bigcup_{t \in V(T_u)} X_t$ and $\bigcup_{t \in V(T_v)} X_t$ form a near-partition of $V(G)$ into two sets. We set

$$\delta^T(e) = \delta_G(\bigcup_{t \in V(T_u)} X_t, \bigcup_{t \in V(T_v)} X_t).$$

The *adhesion* of $(T, \mathcal{X})$ is $\max\{|\delta^T(e)| \mid e \in E(T)\}$.

Given a tree-cut decomposition $(T, \mathcal{X})$ of G and node $t \in V(T)$, let $T_1, \ldots, T_\ell$ be the connected components of $T \setminus t$. The *torso* of G *at* t is a graph obtained from G by identifying each non-empty vertex set $Z_i := \bigcup_{b \in V(T_i)} X_b$ into a single vertex z_i (the multiplicity of the created edges is defined appropriately). For a graph G and a set $X \subseteq V(G)$, the *3-center* of (G, X) is the graph obtained from G by repetitively dissolving every vertex $v \in V(G) \setminus X$ that has two neighbors and degree 2 and removing every vertex $v \in V(G) \setminus X$ of degree at most 2 and at most one neighbor (*dissolving* a vertex x of degree two with exactly two neighbors y and z is the operation of removing x and adding the edge $\{y, z\}$ – if this edge already exists then its multiplicity is increased by one).

Given a tree-cut decomposition $(T, \mathcal{X})$ of G, let H_t be the torso at t and $\bar{H}_t$ be the 3-center of (H_t, X_t). Then the *width* of $(T, \mathcal{X})$ equals

$$\max_{e \in E(T),\ t \in V(T)} \{|\delta^T(e)|, |V(\bar{H}_t)|\}.$$

The tree-cut width of G, or $\mathbf{tcw}(G)$ in short, is the minimum width of $(T, \mathcal{X})$ over all tree-cut decompositions $(T, \mathcal{X})$ of G. In the special case when G is 3-edge-connected, the following observation is not difficult to verify.

Observation 7. *Let G be a 3-edge-connected graph and let $(T, \mathcal{X})$ be a tree-cut decomposition of G. Consider an arbitrary node t of $V(T)$ and let $\mathcal{T}$ be the set containing every connected component T' of $T \setminus t$ such that $\bigcup_{s \in V(T')} X_s \neq \emptyset$. Then $|V(\bar{H}_t)| = |X_t| + |\mathcal{T}|$, that is $|V(\bar{H}_t)| = |V(H_t)|$.*

The following lemma can be easily derived from [30, Lemma 11]. For the sake of completeness, we present a simple proof of it.

Lemma 16. *Given a graph G, let $\{X_1, X_2\}$ be a partition of $V(G)$ such that $|\delta_G(X_1, X_2)| \leq 2$ and let $w \geq 2$ be a positive integer. Then if both $G[X_1]$ and $G[X_2]$ have tree-width at most w, then G has tree-cut width at most w.*

Proof: Let $(T^i, \mathcal{X}^i)$ be a tree-cut decomposition of $G[X_i]$ of width at most w for $i \in [2]$, and consider the tree-cut decomposition $(T, \mathcal{X})$ such that $\mathcal{X} = \mathcal{X}_1 \cup \mathcal{X}_2$ and T is obtained by the disjoint union of T^1 and T^2 after adding an edge with one (arbitrarily chosen) endpoint, say t_1, in T^1 and another (arbitrarily chosen) endpoint, say t_2, in T^2.

We want to prove that $(T, \mathcal{X})$ is a tree-cut decomposition of width at most w. To see this, note first that the adhesion of $(T, \mathcal{X})$ is at most w since $|\delta^T(\{t_1, t_2\})| \leq 2$ and the adhesion of $(T^i, \mathcal{X}^i)$ is at most w for $i \in [2]$. From $|\delta^T(\{t_1, t_2\})| \leq 2$, it follows that for $i \in [2]$, the 3-center of (H_{t_i}, X_{t_i}) of the tree-decomposition $(T, \mathcal{X})$ is the same as the 3-center of (H_{t_i}, X_{t_i}) of the tree-decomposition $(T^i, \mathcal{X}^i)$. Therefore the width of $(T, \mathcal{X})$ is at most w. □

A tree-cut decomposition $(T, \mathcal{X})$ of a graph is *normalized* if all nodes of T with degree at most two are non-trivial.

Observation 8. *If $(T, \mathcal{X})$ is a normalized tree-cut decomposition of G, then $|V(T)| \leq 2 \cdot |V(G)|$.*

Lemma 17. *Let G be a 3-edge connected graph, $w \in \mathbb{Z}_{\geq 2}$ and let $S \subseteq V(G)$ be a set containing at least $w+1$ vertices such that $|\delta_G(S, V(G) \setminus S)| \leq 2w$. Let $I = (H, w, B, w)$ be an instance of the* Special Partitioning *problem such that $H = G[S]$, $B = \partial_G(S)$, and for every $v \in B$, $w(v)$ is the number of edges in $\delta_G(S, V(G) \setminus S)$ that have v as an endpoint. If $\mathbf{tcw}(G) \leq w$, then I is a YES-instance.*

Proof: Let $(T, \mathcal{X})$ be a normalized tree-cut decomposition of G of width at most w. For every edge $e \in E(T)$ with endpoints x and y, let T_x and T_y be the connected components of T which contain the nodes x and y respectively. We extend the weight function w on B into w' on $V(G)$ by setting $w'(v) = w(v)$ for every $v \in B$ and $w'(v) = 0$ otherwise. Also, given a subtree Y of T, we define $w(Y)$ as $\sum_{t \in V(Y)} \sum_{v \in X_t} w(v)$. Consider two rules to orient an edge of T, which may possibly orient an edge in one or two directions, or leave it unoriented. Given an edge $e = \{x, y\}$ of $E(T)$,

Rule 1: orient e from x to y if $w(T_y) > w$.
Rule 2: orient e from x to y if $S \cap \bigcup_{t \in V(T_x)} X_t = \emptyset$.

Let $\boldsymbol{T}$ be the mixed graph obtained from T by applying Rules 1 and 2 exhaustively to the edges of T. We claim that no edge is oriented in two directions.

Claim 6. *For every edge $e = \{x, y\}$ of T, e is oriented either in a single direction or not oriented in $\boldsymbol{T}$.*

Proof of the Claim: Observe that if **Rule 1** orients e from x to y, neither **Rule 1** nor **Rule 2** may orient e in the opposite direction. The former is an immediate consequence of the fact $w(T_x) + w(T_y) = |\delta_G(S, V(G) \setminus S)| \leq 2w$. **Rule 2** does not orient e from y to x either: if **Rule 2** does so, we have $S \cap \bigcup_{t \in V(T_y)} X_t = \emptyset$ and since the value $w(v)$ is non-zero only when $v \in S$, we conclude that $w(T_y) = 0$, a contradiction to the assumption **Rule 1** oriented e from x to y. Moreover, the edge e cannot be oriented in both directions by **Rule 2** since S is non-empty and thus at least one of the sets $\bigcup_{t \in V(T_x)} X_t$ and $\bigcup_{t \in V(T_y)} X_t$ intersects with S. ◇

Since no edge of T is oriented in two directions in $\boldsymbol{T}$, there is at least one node, say t, which does not have an out-going edge in $\boldsymbol{T}$. Let $T_1, \ldots, T_\ell$ be the connected components of $T \setminus t$. Notice that $\ell \geq 1$, as $|V(G)| \geq |S| \geq w+1$. Consider the following $(r+1)$-allocation $\mathcal{V}$ of $V(H) = S$, where $r := \ell$.

$$\mathcal{V}^{(i)} = \begin{cases} S \cap \bigcup_{b \in V(T_i)} X_b & \text{for all } i \in [r] \\ S \cap X_t & \text{for } i = r+1 \end{cases}$$

It remains to show that $\mathcal{V}$ is a solution to Special Partitioning for $I = (H, w, B, w)$. For this we verify that $\mathcal{V}$ satisfies the four conditions for an $(r+1)$-allocation to be a solution to Special Partitioning. For Condition 1, suppose

to the contrary that at most one part of $\mathcal{V}$ is non-empty. Since S is a non-empty set, this means that there exists exactly one part, say $\mathcal{V}^{(i)}$ for some $i \in [r+1]$, which is non-empty. Observe that $i \neq r+1$ since otherwise, $S = \mathcal{V}^{(r+1)} \subseteq X_t$ and X_t contains more than w vertices, contradicting to the assumption that $(T, \mathcal{X})$ is a tree-cut decomposition of width at most w. Consider the edge $e = \{t_i, t\}$, where t_i is the neighbor of t corresponding to the subtree T_i. Due to the assumption that $\bigcup_{j \neq i} \mathcal{V}^{(j)} = \emptyset$, we know that e is oriented from t to t_i in $\boldsymbol{T}$, which is a contradiction to the assumption that t does not have an out-going edge in $\boldsymbol{T}$.

Regarding Condition 2, note that $|\mathcal{V}^{(r+1)}| + r \leq |X_t| + \ell$. Since $(T, \mathcal{X})$ is a normalized tree-cut decomposition, we have $\bigcup_{b \in V(T_i)} X_b \neq \emptyset$ for every $i \in [\ell]$. Hence, Observation 7 and the fact that the width of $(T, \mathcal{X})$ is at most w implies that $|X_t| + \ell \leq w$ and thus, Condition 2 is satisfied by $\mathcal{V}$. Condition 3 is also satisfied since t does not have an out-going edge in $\boldsymbol{T}$, in particular, **Rule 1** does not orient any edge incident with t outwardly from t.

Lastly, observe that for every $i \in [r]$, we have

$$|\delta_H(\mathcal{V}^{(i)}, V(H) \setminus \mathcal{V}^{(i)})| \leq |\delta_G(\bigcup_{b \in V(T_i)} X_b, V(G) \setminus \bigcup_{b \in V(T_i)} X_b)| \leq w,$$

where the second inequality follows from the fact that the width, and thus the adhesion, of $(T, \mathcal{X})$ is at most w. This completes the proof of our claim that $\mathcal{V}$ is a solution to Special Partitioning for $I = (H, w, B, w)$, and thus the proof of the statement. □

Let G be 3-edge connected graph and let $(T, \mathcal{X})$ be a tree-cut decomposition of G with at least two nodes. The *internal-width* of $(T, \mathcal{X})$ equals

$$\max_{e \in E(T),\ t \in V(T) \setminus L(T)} \{|\delta^T(e)|, |V(H_t)|\}.$$

We also define $B_w(T, \mathcal{X}) = V(T) \setminus \{t \in L(T) \mid |X_t| \geq w\}$.

Observation 9. *Let $w \in \mathbb{Z}_{\geq 2}$. If $(T, \mathcal{X})$ is a tree-cut decomposition of a 3-edge connected graph G with internal-width at most w and where $B_w(T, \mathcal{X}) = |V(T)|$ then $\mathbf{tcw}(G) \leq w$.*

Lemma 18. *There exists an algorithm that, given $w \in \mathbb{Z}_{\geq 1}$ and a graph G where $|V(G)| > w$, either outputs that $\mathbf{tcw}(G) > w$ or outputs a normalized tree-cut decomposition of G with at least two nodes and internal width at most $2w$. The algorithm runs in $O(n^4)$ steps.*

Proof: The algorithm first computes a partition $\{X_1, X_2\}$ of $V(G)$ such that $|\delta_G(X_1, X_2)|$ is minimum using the $O(n^3)$ algorithm in [28].

Notice that if $\mathbf{tcw}(G) \leq w$ and $|V(G)| > w$, then every tree-cut decomposition $(T, \mathcal{X})$ of G of width at most w contains an edge e of T such that $|\delta^T(e)| \leq w$. Therefore, if $|\delta_G(X_1, X_2)| > w$, then the algorithm can safely report that $\mathbf{tcw}(G) > w$. Otherwise the algorithm outputs $(T, \mathcal{X})$ where $T = (\{x_1, x_2\}, \{\{x_1, x_2\}\})$ and $\mathcal{X} = \{X_1, X_2\}$. The lemma follows as $(T, \mathcal{X})$ has internal-width at most $w \leq 2w$. □

Algorithm: **build-tcd**$(G, T, \mathcal{X}, w)$
Input : A 3-edge connected graph G and normalized tree-cut decomposition $(T, \mathcal{X})$ of G of at least two nodes and internal-width at most $2w$ and $B_{2w}(T, \mathcal{X}) \neq V(T)$, where $w \in \mathbb{Z}_{\geq 2}$.
Output : Either a normalized tree-cut decomposition $(T', \mathcal{X}')$ of G of at least two nodes and internal-width at most $2w$ and where $|B_{2w}(T, \mathcal{X})| < |B_{2w}(T', \mathcal{X}')|$ or a report that $\mathbf{tcw}(G) > w$.

1. **let** t be some node in $T(V) \setminus B_{2w}(T, \mathcal{X})$
2. **let** $I_t = (G[X_t], w, \partial_G(X_t), w)$, where for every $x \in \partial_G(X_t)$, $w(x)$ is the number of edges in $\delta_G(X_t, V(G) \setminus X_t)$ that contain x as an endpoint.
3. **if** I_t is a NO-instance of Special Partition then return that $\mathbf{tcw}(G) > w$, otherwise, let $\mathcal{V}$ the allocation of X_t created from a solution $\mathcal{V}^*$ for I_t after discarding from $\mathcal{V}^*$ every empty part that is not the central one (notice that $\mathcal{V}$ is again a solution for I_t).
4. **let** $(T', \mathcal{X}')$ be a tree-cut decomposition such that T' is obtained from T after adding r new nodes $t_1, \ldots, t_r$ in it (here, $r = |\mathcal{V}|$).
5. **let** $\mathcal{X}' = \{X'_b \mid b \in V(T')\}$ be such that $X'_b = X_b$ for $b \in V(T) \setminus \{t\}$, $X'_t = \mathcal{V}^{r+1}$, and $X'_{t_i} = \mathcal{V}^{(i)}$ for $i \in [r]$.
6. **return** $(T', \mathcal{X}')$

Lemma 19. *The algorithm* **build-tcd** *is correct and runs in* $2^{O(w^4 \cdot \log w)} \cdot n^4 \cdot \log n)$ *steps.*

Proof: If in Line 3 the I_t is a NO-instance of Special Partition then by Lemma 17, the algorithm correctly reports that $\mathbf{tcw}(G) > w$. Suppose now that I_t is a YES-instance of Special Partition. Clearly the tree-cut decomposition $(T', \mathcal{X}')$ constructed in Lines 4 and 5 contains at least two nodes. To prove that $(T', \mathcal{X}')$ is normalized and $|B_{2w}(T, \mathcal{X})| < |B_{2w}(T', \mathcal{X}')|$, observe that $\mathcal{V}^*$ contains at least two non-empty sets, therefore either the central part of $\mathcal{V}$ is empty and $|\mathcal{V}| \geq 3$ or the central part of $\mathcal{V}$ is non-empty and $|\mathcal{V}| \geq 2$.

It now remains to prove that the internal-width of $(T', \mathcal{X}')$ is at most $2w$. Since $\mathcal{V}$ is a solution to I_t, the size of the torso at t in $(T', \mathcal{X}')$ is $|V(H_t)| = |X'_t| + r \leq |\mathcal{V}^{(r+1)}| + r \leq w$. Let us verify that the adhesion of $(T', \mathcal{X}')$ is at most $2w$. For this, it suffices to bound the value $|\delta^{T'}(e)|$ for the newly created edges $e = \{t_i, t\}$, for all $i \in [r]$. Notice first that $\{X_t \setminus X'_{t_i}, V(G) \setminus X_t\}$ is a partition of $V(G) \setminus X'_{t_i}$. We have

$$\begin{aligned} |\delta^{T'}(\{t_i, t\})| &= |\delta_G(X'_{t_i}, V(G) \setminus X'_{t_i})| \\ &= |\delta_G(X'_{t_i}, X_t \setminus X'_{t_i})| + |\delta_G(X'_{t_i}, V(G) \setminus X_t)| \\ &= |\delta_G(\mathcal{V}^{(i)}, X_t \setminus \mathcal{V}^{(i)})| + w(\partial_G(X_t) \cap \mathcal{V}^{(i)}) \leq 2w, \end{aligned}$$

where the last inequality follows from that $\mathcal{V}$ is a solution for I_t.

Add the running time of the algorithm is dominated by Step **3**, the claimed running time follows. □

Lemma 20. *There exists an algorithm that, given a 3-edge-connected graph G and a $w \in \mathbb{Z}_{\geq 0}$, either outputs a tree-cut decomposition of G with width at most $2w$ or reports that no tree-cut decomposition of G with width at most w exists in $2^{O(w^4 \cdot \log w)} \cdot n^5 \cdot \log n$ steps.*

Proof: We assume that $w \geq 2$ as otherwise we can directly return a negative answer (recall that if G is 3-edge connected then $\mathbf{tcw}(G) \geq 2$). We also assume that $|V(G)| > w$, otherwise the algorithm directly outputs that $\mathbf{tcw}(G) \leq w$. The algorithm applies Lemma 18 and, if the output is a tree-cut decomposition $(T, \mathcal{X})$, then repetitively applies **build-tcd** on $(G, T, \mathcal{X})$ as long as it receives new tree-cut decomposition $(T', \mathcal{X}')$ where $B_{2w}(T', \mathcal{X}') \neq |V(T')|$. This procedure ends either when **build-tcd** correctly reports that $\mathbf{tcw}(G) > w$ or produces a normalized tree-cut decomposition $(T', \mathcal{X}')$ of G of at least two nodes and internal-width at most $2w$, where $|B_{2w}(T', \mathcal{X}')| = |V(T')|$.

Because of Lemma 19, in the first case the algorithm correctly concludes that $\mathbf{tcw}(G) > w$ and in the second case it outputs a tree-cut decomposition of width at most $2w$, because of Observation 9.

By the same Lemma 19, the output of **build-tcd** is always normalized and $|B_{2w}(T, \mathcal{X})| < |B_{2w}(T', \mathcal{X}')|$. This together with Observation 8, implies that the algorithm calls **build-tcd** $O(n)$ times. □

Algorithm: **apr-tcw**(G, w)
Input : A graph G where $|V(G)| \geq 1$ and $w \in \mathbb{Z}_{\geq 1}$.
Output : Either a tree-cut decomposition $(T, \mathcal{X})$ of G with width at most $2w$ or a report that $\mathbf{tcw}(G) > w$.

1. **if** $|V(G)| = 1$, then **return** $((V(G), \emptyset), V(G))$
2. find a partition $\{X_1, X_2\}$ of $V(G)$ such that $\delta = |\delta_G(X_1, X_2)|$ is minimum
3. **if** $\delta \geq 3$, then **return** the answer of the algorithm of Lemma 20 applied for G and w.
4. **if**, for $i \in [2]$, the output of **apr-tcw**$(G[X_i], w)$ is a tree-cut decomposition $(T^i, \mathcal{X}^i)$ of $G[X_i]$ with width at most $2w$ then
5. **return** the tree-cut decomposition $(T, \mathcal{X})$ such that T is obtained by the disjoint union of T^1 and T^1
6. after adding an edge with one (arbitrarily chosen) endpoint in T^1 and one (arbitrarily chosen)
7. endpoint in T^2 and $\mathcal{X} = \mathcal{X}^1 \cup \mathcal{X}^2$.
8. **return** that $\mathbf{tcw}(G) > w$.

Theorem 5. *There exists an algorithm that, given a graph G and a $w \in \mathbb{Z}_{\geq 0}$, either outputs a tree-cut decomposition of G with width at most $2w$ or correctly reports that no tree-cut decomposition of G with width at most w exists in $2^{O(w^4 \cdot \log w)} \cdot n^5 \cdot \log n$ steps.*

PROOF: The algorithm is the following: If $V(G) = \emptyset$, then the algorithm returns $\{(\{x\}, \emptyset), \{\emptyset\}\}$. Therefore, we may assume that G contains at least one vertex. If $w = 0$, then the algorithm returns that $\mathbf{tcw}(G) > w$. If $w = 1$, the algorithm does the following: it returns $\mathbf{tcw}(G) > w$ if G contains a cycle and otherwise, i.e., G is a forest, outputs $((V(G), E(G)), \{\{u\} : u \in V(G)\})$. If now $|V(G)| \geq 1$ and $w \geq 2$, then the algorithm returns $\mathbf{apr\text{-}tcw}(G, w)$.

First, we prove the correctness of the algorithm. The correctness is trivially verified when $V(G) = \emptyset$ or $w \leq 1$. Hence we consider the case when $|V(G)| \geq 1$ and $w \geq 2$ and show that $\mathbf{apr\text{-}tcw}(G, w)$ either outputs a tree-cut decomposition of G with width at most $2w$ or correctly reports $\mathbf{tcw}(G) > w$. We observe that the algorithm $\mathbf{apr\text{-}tcw}(G, w)$ iteratively decomposes $V(G)$ and recursively calls $\mathbf{apr\text{-}tcw}(G, w)$ until either $|V(G)| = 1$ or G is 3-edge-connected. In case $|V(G)| = 1$, it is trivial to see that Line 1 returns a tree-cut decomposition with width at most $2w$. In case when G is 3-edge-connected, $\mathbf{apr\text{-}tcw}(G, w)$ returns an output in Line 3, whose correctness is guaranteed by Lemma 20. Hence, we proceed the correctness proof by induction on $|V(G)|$ and assume that $\mathbf{apr\text{-}tcw}(G, w)$ returns a correct output for all input graphs whose number of vertices is strictly smaller than $|V(G)|$.

If $\mathbf{apr\text{-}tcw}(G, w)$ returns a tree-cut decomposition $(T, \mathcal{X})$ in Line 5, then by induction hypothesis, $(T^i, \mathcal{X}^i)$ is a tree-cut decomposition of $G[X_i]$ with width at most $2w$, for $i \in [2]$. By Lemma 16 and from $w \geq 2$, it follows that $(T, \mathcal{X})$ has width at most $2w$. If $\mathbf{apr\text{-}tcw}(G, w)$ reports that $\mathbf{tcw}(G) > w$ in Line 8, it means that the tree-cut width of $G[X_i]$ is reported to be strictly larger than w for some $i \in [2]$, which is correct by induction hypothesis. Since the tree-cut width is closed under taking a subgraph, it follows that $\mathbf{tcw}(G) > w$ and thus the output in Line 8 is correct. This complete the proof of correctness.

The claimed running time is an immediate consequence of the running time claimed in Lemma 20, the fact that $|E(G)| = O(w^3 \cdot |V(G)|)$, and the following recursion: $T(n, w) \leq T(n_1, w) + T(n_2, w) + O(n)$, where n_i is the size $|X_i|$ of the input graph for the subroutine called in Line 4 for $i \in [2]$. The additive factor $O(n)$ is the running time consumed to compute $(T, \mathcal{X})$ from $(T^i, \mathcal{X}^i)$ for $i \in [2]$ in Line 5. □