Parameterized Algorithms for Min-Max Multiway Cut and List Digraph Homomorphism*†

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

1 **CNRS, LAMSADE, Paris, France**
`eunjungkim78@gmail.com`
2 **AlGCo project-team, CNRS, LIRMM, Montpellier, France**
`{Christophe.Paul,Ignasi.Sau}@lirmm.fr`, `sedthilk@thilikos.info`
3 **Department of Mathematics, University of Athens, Athens, Greece**
4 **Computer Technology Institute & Press "Diophantus", Patras, Greece**

Abstract

In this paper we design FPT-algorithms for two parameterized problems. The first is List Digraph Homomorphism: given two digraphs G and H and a list of allowed vertices of H for every vertex of G, the question is whether there exists a homomorphism from G to H respecting the list constraints. The second problem is a variant of Multiway Cut, namely Min-Max Multiway Cut: given a graph G, a non-negative integer ℓ, and a set T of r terminals, the question is whether we can partition the vertices of G into r parts such that (a) each part contains one terminal and (b) there are at most ℓ edges with only one endpoint in this part. We parameterize List Digraph Homomorphism by the number w of edges of G that are mapped to non-loop edges of H and we give a time $2^{O(\ell \cdot \log h + \ell^2 \cdot \log \ell)} \cdot n^4 \cdot \log n$ algorithm, where h is the order of the host graph H. We also prove that Min-Max Multiway Cut can be solved in time $2^{O((\ell r)^2 \log \ell r)} \cdot n^4 \cdot \log n$. Our approach introduces a general problem, called List Allocation, whose expressive power permits the design of parameterized reductions of both aforementioned problems to it. Then our results are based on an FPT-algorithm for the List Allocation problem that is designed using a suitable adaptation of the *randomized contractions* technique (introduced by [Chitnis, Cygan, Hajiaghayi, Pilipczuk, and Pilipczuk, FOCS 2012]).

1998 ACM Subject Classification G.2.1 Combinatorics, G.2.2 Graph Theory

Keywords and phrases Parameterized complexity, Fixed-Parameter Tractable algorithm, Multiway Cut, Digraph homomorphism

Digital Object Identifier 10.4230/LIPIcs.IPEC.2015.78

1 Introduction

The Multiway Cut problem asks, given a graph G, a set of r terminals T, and a non-negative integer ℓ, 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 ℓ edges between different parts (i.e., at most ℓ crossing edges). In the special case where $|T| = 2$, this gives the celebrated Minimum Cut problem, which is polynomially solvable [31]. In general, when there is no

* The third author was co-financed by the European Union (European Social Fund ESF) and Greek national funds through the Operational Program "Education and Lifelong Learning" of the National Strategic Reference Framework (NSRF), Research Funding Program: ARISTEIA II.
† A complete version of this extended abstract has been appeared at http://arxiv.org/abs/1509.07404

restriction on the number of terminals, the MULTIWAY CUT problem is NP-complete [8] 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 [18, 3, 14, 17, 24, 30].

More recently, special attention to the MULTIWAY CUT problem was given from the parameterized complexity point of view. The existence of an FPT-algorithm for MULTIWAY CUT (when parameterized by ℓ), i.e., an $f(\ell) \cdot n^{O(1)}$-step algorithm, had been a long-standing open problem. This question was answered positively by Marx in [26] with the use of the *important separators technique* which was also used for the design of FPT-algorithms for several other problems such as DIRECTED MULTIWAY CUT [4], VERTEX MULTICUT, and EDGE MULTICUT [28]. This technique has been extended to the powerful framework of *randomized contractions technique*, introduced in [5]. This made it possible to design FPT-algorithms for several other problems such as UNIQUE LABEL COVER, STEINER CUT, EDGE/VERTEX MULTIWAY CUT-UNCUT. We stress that this technique is quite versatile.

In this paper we use it in order to design FPT-algorithms for parameterizations of two problems that do not seem to be directly related to each other: the MIN-MAX-MULTIWAY CUT problem [32] and the LIST DIGRAPH HOMOMORPHISM problem.

1.1 Min-Max-Multiway Cut

In the MULTIWAY CUT problem the parameter ℓ bounds the total number of crossing edges (i.e., edges with endpoints in different parts). Svitkina and Tardos [32] considered a “min-max” variant of this problem, namely the MIN-MAX-MULTIWAY CUT, where ℓ bounds the maximum number of outgoing edges of the parts[1]. In [32], it was proved that MIN-MAX-MULTIWAY CUT is NP-complete even when the number of terminals is $r = 4$. As a consequence of the results in [32] and [29], 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. We prove the following.

▶ **Theorem 1.** *There exists an algorithm that solves the* MIN-MAX-MULTIWAY CUT *problem in* $2^{O((r\ell)^2 \log r\ell)} \cdot n^4 \cdot \log n$ *steps, i.e.,* MIN-MAX-MULTIWAY CUT *belongs to FPT when parameterized by both* r *and* ℓ*.*

(Throughout the paper, we use $n = |V(G)|$ when we refer to the number of vertices of the graph G in the instance of the considered problem.)

1.2 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 H-homomorphism such that for every vertex v of G, $\chi(v) \in \lambda(v)$. Graph and digraph homomorphisms have been extensively studied both from the combinatorial and the algorithmic point of view (see e.g., [21, 2, 13, 15, 16]).

Especially for the LIST DIGRAPH HOMOMORPHISM problem, a dichotomy characterizing the instantiations of H for which the problem is hard was given in [22] (see also [12]). Notice that the standard parameterization of LIST DIGRAPH HOMOMORPHISM by the size of the

[1] Notice that under this viewpoint MULTIWAY CUT can be seen as MIN-SUM-MULTIWAY CUT.

graph H is para-NP-complete as it yields the 3-Coloring problem when G is restricted to be a simple graph and $H = K_3$. A more promising 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 [9, 11, 27]). Another parameterization, again for the undirected case, was introduced in [6], 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.

We introduce a new parameterization of List Digraph Homomorphism where the parameter is, apart from $h = |V(H)|$, the number of "crossing edges", i.e., the edges of G whose endpoints are mapped to different vertices of H. For this, we enhance the input with an integer ℓ and ask for a list digraph homomorphism with at most ℓ crossing edges. Clearly, when $\ell = |E(G)|$, this yields the original problem. We call the new problem Bounded List Digraph Homomorphism (in short, BLDH). Notice that the fact that List Digraph Homomorphism is NP-complete even when $h = 3$, implies that BLDH is para-NP-complete when parameterized only by h. The input of BLDH is a quadruple (G, H, λ, ℓ) where G is the guest graph, H is the host graph, $\lambda : V(G) \to 2^{V(H)}$ is the list function and ℓ is a non-negative integer. Our next step is to observe that BLDH is W[1]-hard, when parameterized only by ℓ. To see this consider an input (G, k) of the Clique problem and construct the input $(K, \bar{G}, \lambda, \ell)$ where K is a the complete digraph on k vertices, $\bar{G}$ is the digraph obtained by G by replacing each edge by two opposite direction arcs between the same endpoints, $\lambda = \{(v, V(G)) \mid v \in V(K)\}$, and $\ell = k(k-1)$. Notice that (G, k) is a yes-instance of Clique iff $(K, \bar{G}, \lambda, \ell)$ is a yes-instance of BLDH.

We conclude that when BLDH is parameterized by ℓ or h only, then one may not expect it to be fixed parameter tractable. This means that the parameterization of BLDH by h and ℓ is meaningful to consider. Our result is the following.

▶ **Theorem 2.** *There exists an algorithm that solves the* Bounded List Digraph Homomorphism *problem in* $2^{O(\ell \cdot \log h + \ell^2 \cdot \log \ell)} \cdot n^4 \cdot \log n$ *steps, i.e.,* Bounded List Digraph Homomorphism *belongs to FPT when parameterized by the number ℓ of crossing edges and the number h of vertices of H.*

1.3 List Allocation

In order to prove Theorems 1 and 2, we prove that both problems are Turing FPT-reducible[2] to a single new problem that we call List Allocation (in short, LA).

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* $\alpha(i, j)$.

As we easily see in Subsection 2.3, 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., $w = \sum_{1 \leq i < j \leq r} \alpha(i, j)$.

[2] 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 "T-FPT" instead of "Turing FPT".)

Our main result is that this parameterization of LA is in FPT (the basic ideas of the algorithm are given in Section 4.

▶ **Theorem 3.** *There exists an algorithm that, given as input an instance $I = (G, r, \lambda, \alpha)$ of* List Allocation, *returns an answer to this problem in $2^{O(w^2 \cdot \log w)} \cdot n^4 \cdot \log n$ steps, where $w = \sum_{1 \leq i < j \leq r} \alpha(i, j)$.*

To witness the expressive power of List Allocation, let us first exemplify why Multiway Cut, parameterized by w, is T-FPT-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 $(G, T = \{t_1, \dots, t_r\}, w)$ where $r \leq w + 1$.

Next, we consider the set $\mathcal{A}$ containing every weight function α such that $\sum_{1 \leq i < j \leq r} \alpha(i, j) \leq w$. Let also $\lambda : V(G) \to 2^{[r]}$ be the list function such that if $v = t_i \in T$, then $\lambda(v) = \{i\}$, otherwise $\lambda(v) = \{1, \dots, r\}$. 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. This reduction to the List Allocation problem turns out to be quite flexible and, as we will see in Subsection 3.1 (Theorem 4), it can easily be adapted to a T-FPT-reduction of Min-Max-Multiway Cut to List Allocation. The reduction of Bounded List Digraph Homomorphism to List Allocation is more complicated and is described in Subsection 3.2 (Theorem 10). Theorem 3, together with the aforementioned reductions, yields Theorems 1 and 2.

2 Preliminaries and the definition of List Allocation

2.1 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 a function $f : A \to \mathbb{Z}_{\geq 0}$ we define $\sum f = \sum_{x \in A} f(x)$. An *$r$-allocation* of a set S is an r-tuple $\mathcal{V} = (V_1, \dots, V_r)$ 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., $\mathcal{V}^{(i)} = V_i$.

2.2 Definitions about graphs

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 $m = |E(G)|$.

All graphs in this paper are loopless and they may have multiple edges. The only exception to this agreement is in Subsection 3.2 where we also allow loops. 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.

2.3 The list allocation problem

We define the problem LA as follows.

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

For simplicity, in the above definition we write $\alpha(\{i,j\})$ as $\alpha(i,j)$ and we agree that $\alpha(i,j) = \alpha(j,i)$. Also, given an instance I of LA, we denote[3] $w(I) = \sum \alpha$. We will also use w instead of $w(I)$ when it is clear what is the instance we are working with. We assume that the multiplicity of each edge in G does not exceed w as, if this happens, then reducing it to w creates an equivalent instance of the problem.

In the definition of LA 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 precise number of crossing edges between distinct parts of $\mathcal{V}$.

Notice that LA is an NP-hard problem by a simple reduction from the Max Cut problem, asking whether, for an input graph G and some $w \in \mathbb{Z}_{\geq 0}$, whether there is a partition V_1, V_2 of $V(G)$ such that there are *exactly*[4] w edges each with endpoints in both V_1 and V_2. Indeed, given an instance $I = (G, w)$ of Max Cut, construct the instance $I' = (G, 2, \lambda, \alpha)$ where $\lambda(v) = \{1, 2\}$ for every $v \in V(G)$ and $\alpha(1,2) = w$. Note also that when $r = 2$, LA is polynomially solvable on planar graphs as it directly reduces to Planar Max Cut that is polynomially solvable [20].

3 Main reductions

In this section we formally define Min-Max-Multiway Cut and List Digraph Homomorphism and we reduce them to List Allocation.

3.1 Min-Max-Multiway Cut

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

Min-Max-Multiway Cut
Input: A tuple $I = (G, \ell, r, T)$ where G is an undirected graph, $\ell, r \in \mathbb{Z}_{\geq 0}$, and $T \subseteq V(G)$ with $|T| = r$.
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 $|\delta_G(\mathcal{P}_i, V(G) \setminus \mathcal{P}_i)| \leq \ell$, or a correct report that no such partition exists.

Similarly to the case of LA, we assume that the multiplicity of each edge in G does not exceed ℓ.

▶ **Theorem 4.** *If there is an algorithm that solves* LA *in $T(n, w(I))$ steps, then there exists an algorithm that solves* Min-Max-Multiway Cut *in $2^{O(r \cdot \min\{\ell \cdot \log r, r \cdot \log \ell\})} \cdot T(n, r\ell)$ steps.*

[3] Given a function $\tau : A \to \mathbb{Z}_{\geq 0}$, we denote $\sum \tau = \sum_{x \in A} \tau(x)$.
[4] It is straightforward to see that the standard reduction from Nae-3-Sat also works when the question of Max Cut asks for exactly w crossing edges instead of at least w crossing edges.

Proof. Given an input $I = (G, \ell, r, T)$ of MIN-MAX-MULTIWAY CUT, we fix (arbitrarily) 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 consider the family $\mathcal{U}(I)$ of instances of LA containing one element $I' = (G, r, \lambda, \alpha)$ for each choice of function $\alpha : \binom{[r]}{2} \to \mathbb{Z}_{\geq 0}$ satisfying

$$\forall i \in [r], \sum_{j \in [r] \setminus i} \alpha(i, j) \leq \ell.$$

Notice that I is a YES-instance of MIN-MAX-MULTIWAY CUT if and only if there exists some $I' \in \mathcal{U}(I)$ that is a YES-instance of LA. As $|\mathcal{U}(I)| = 2^{O(r \cdot \min\{\ell \cdot \log r, r \cdot \log \ell\})}$ and for each $I' \in \mathcal{U}(I)$ it holds that $w(I') = O(r\ell)$, the result follows. ◀

3.2 List Digraph Homomorphism

Let G and H be directed graphs where G is simple and H may have loops but not multiple directed edges. A (directed) edge in the digraph G from the vertex x to the vertex y is denoted by (x, y). 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

- $\chi(v) \in \lambda(v)$ for every $v \in V(G)$, and
- $(\chi(u), \chi(v)) \in E(H)$ for every $(u, v) \in E(G)$.

Given a list H-homomorphism χ of G and an edge $e = (a, b) \in E_2(H)$ we define

$$C(e) = \{(u, v) \in E(G) \mid \chi(u) = a \text{ and } \chi(v) = b\}.$$

BOUNDED LIST DIGRAPH HOMOMORPHISM is formally defined as follows.

BOUNDED LIST DIGRAPH HOMOMORPHISM (BLDH)
Input: A tuple $I = (G, H, \lambda, \ell)$ where G and H are digraphs, $\lambda : V(G) \to 2^{V(H)}$, and $\ell \in \mathbb{N}_{\geq 0}$.
Output: A *λ-list H-homomorphism* of G where $\sum_{e \in E(H)} |C(e)| \leq \ell$ or a correct report that no such homomorphism exists.

We now define the following more general problem.

ARC-SPECIFIED LIST DIGRAPH HOMOMORPHISM (ASLDH)
Input: A tuple $I = (G, H, \lambda, \alpha)$ where G and H are digraphs, $\lambda : V(G) \to 2^{V(H)}$, and $\alpha : E_2(H) \to \mathbb{Z}_{\geq 0}$.
Output: A λ-list H-homomorphism χ of G such that $\forall_{e \in E_2(H)}\ |C(e)| = \alpha(e)$ or a correct report that no such λ-list H-homomorphism exists.

Given an instance $I = (G, H, \lambda, \alpha)$ of ASLDH we define $d(I) = \sum \alpha$. As we already did for the cases of LA and MIN-MAX-MULTIWAY CUT, we assume that the multiplicity of the edges of the instance of BLDH (resp. ASLDH) does not exceed ℓ (resp. $d(I)$).

In the next sections we will prove that there exists an FPT-algorithm for ASLDH, when parameterized by *both* $h = |V(H)|$ and $d = d(I)$. This fact together with the following result yields Theorem 2.

▶ **Theorem 5.** *If there is an algorithm that solves* ASLDH *in $T(n, d(I))$ steps, then there exists an algorithm that solves* BLDH *in $2^{O(\ell \log h)} \cdot T(n, \ell)$ steps where $h = |V(H)|$.*

Proof. Given an instance $I = (G, H, \lambda, \ell)$ of BLDH we set $\mathcal{U}(I) = \{(G, H, \lambda, \alpha) \mid \sum \alpha \leq \ell\}$ and we observe that I is a YES-instance of BLDH if and only if some $I' \in \mathcal{U}(I)$ is a YES-instance of ASLDH. The lemma follows as $|\mathcal{U}(I)| = 2^{O(\ell \log h)}$ and $d(I') \leq \ell$. ◀

3.3 A sparsifier for ASLDH

In order to prove that ASLDH admits an FPT-algorithm when parameterized by *both* $h = |V(H)|$ and $d = d(I)$, we will give a Turing-FPT reduction of ASLDH to LA in Subsection 3.4. The latter problem can be solved by an FPT-algorithm due to the result of Section 4. The reduction of Subsection 3.4 receives an instance (G, H, λ, α) of ASLDH and returns an equivalent instance $(G', r, \lambda', \alpha')$ of LA where $|V(G')| = O(|E(G)|)$ which is $O(w \cdot |V(G)|^2)$, in general. In order to avoid this blow-up in the polynomial running time of our final FPT-algorithm, we give a way to transform the instances of ASLDH to equivalent instances of the same problem whose graphs are sparse. This "sparsification" procedure is described below.

A graph is *d-edge connected* if it has at least two vertices and for every two vertices there are d edge disjoint paths between them. We use the following result from [25].

▶ **Proposition 6.** *For every $d \in \mathbb{Z}_{\geq 1}$, every graph G where $|E(G)| \geq d \cdot (|V(G)| - 1)$ contains a d-edge connected subgraph.*

We also need the following result.

▶ **Lemma 7.** *Let G be a graph and let $\mathcal{C} = \{C_1, \ldots, C_r\}$ be a collection of vertex disjoint connected subgraphs of G. Let also G' be the graph obtained if we contract in G all edges in the graphs in $\mathcal{C}$. If G' is d-edge connected and each graph in $\mathcal{C}$ is d-edge connected or a single vertex, then G contains a subgraph that is d-edge connected.*

Given a graph H and a positive integer d, we say that a subgraph H of G is a *d-edge connected core of* G if every connected component of H is d-edge connected and, among all such subgraphs of G, H has maximum number of edges. The proof of the next lemma uses Proposition 6.

▶ **Lemma 8.** *For every $d \in \mathbb{Z}_{>0}$, every graph G with $m \geq d \cdot (n-1)$ contains a unique d-edge connected core that can be found in $O(d \cdot n^4)$ steps.*

Proof. The claimed d-edge connected core exists because of Proposition 6. Also, it is unique because if there are two d-edge connected cores J_1 and J_2, then it can be easily checked that the graph $J_1 \cup J_2$ is also a d-edge connected core of G. The algorithm repetitively removes from G edges of min-cuts of size at most $d - 1$ in its connected components (each can be found in $O(d \cdot n^3)$ steps according to [31]) until this is not possible anymore (isolated vertices, when appearing during this procedure, are removed).

Note that the total number of steps of this procedure is bounded by the running time of the algorithm in [31] times the number of connected components of the resulting graph. This justifies the claimed running time. Let J be the d-edge connected core of G. Notice that none of the edges of J will be deleted by this procedure. Indeed, assuming the opposite, let G' be the graph where for the first time a cut (V_1, V_2) is found where the set F of crossing edges contains some edge $e = \{x, y\}$ in J. Let also C be the connected component of G' containing this cut and let C_J be a connected component of J that is a subgraph of C containing e.

Notice that x and y belong to different connected components of $C \setminus F$ and therefore also to different connected components of $C_J \setminus F$, contradicting the fact that C_J is d-edge connected. We just proved that the output of the algorithm will be a subgraph of J. Notice also that each connected component of this output is d-edge connected. By the maximality of J, this output is necessarily J. ◀

▶ **Lemma 9.** *There is an $O(d(I) \cdot n^4)$-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 $|E(G')| = O(d(I) \cdot |V(G')|)$.*

Proof. Let $\tilde{G}$ be the underlying graph of G (multiplicities of edges of opposite direction are summed up) and $d = d(I)$. If $\tilde{G}$ does not contains a $(d+1)$-edge connected core, then, from Proposition 6, $|E(G)| = O(d \cdot |V(G)|)$.

Suppose now that $\tilde{G}$ has a $(d+1)$-edge connected core J that, from Lemma 8, can be found in $O(d \cdot |V(G)|^4)$ steps. We create a new graph G' as follows: for each $C \in \mathcal{C}(J)$ we contract all vertices of C to a single vertex v_C and we update λ to λ' so that if $x \notin \{v_C \mid C \in \mathcal{C}(J)\}$, then $\lambda'(x) = \lambda(x)$ and if $x = v_C$, then $\lambda'(x) = \cap_{y \in V(C)} \lambda(y)$. We claim that $I' = (G', H, \lambda', \alpha)$ is an equivalent instance of ASLDH. Indeed, this is based on the fact that, given a λ-list H-homomorphism χ of G and a connected component C of J, all vertices of J should be the preimages via χ of the same vertex of H. To verify this fact, just observe that, if this is not the case, then the removal of the $\leq d$ crossing edges from C (i.e., edges with endpoints mapped to different vertices of H) will disconnect C, a contradiction to the $(d+1)$-edge-connectivity of C.

It now remains to prove that $|E(G')| = O(d \cdot |V(G')|)$. If $|E(G')| \geq (d+1) \cdot (|V(G)'| - 1)$, then, again from Proposition 6, G' contains a $(d+1)$-edge connected subgraph. This, because of Lemma 7, implies that G contains a subgraph that is $(d+1)$-edge connected and has more edges than J, a contradiction. ◀

3.4 A reduction of ASLDH to LA

Given the results of the previous section we are now in position to prove the following.

▶ **Theorem 10.** *If there is an algorithm that solves* LA *in $T(n, w(I))$ steps, then there exists an algorithm that solves* ASLDH *in $T\big(O(d(I) \cdot n), O(d(I))\big) + O(d(I) \cdot n^4)$ steps.*

Proof. Let $I = (G, H, \lambda, \alpha)$ be an instance of ASLDH. Using the algorithm of Lemma 9, we may assume that $|E(G)| = O(d(I) \cdot |V(G)|)$. We then use I to 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 $\sigma : V(\tilde{H}) \to [r]$ is a 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

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

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

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

Let $\chi : V(G) \to V(H)$ be a λ-list H-homomorphism of G where $\forall_{e \in E_2(H)}\ |C(e)| = \alpha(e)$. 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

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

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

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

It is easy 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 that $\chi(u) = \sigma^{-1}(i)$ if and only if $u \in \mathcal{V}^{(i)}$. We claim that χ is a λ-list H-homomorphism of G where $\forall_{e \in E_2(H)}\ |C(e)| = \alpha(e)$. For this, we investigate χ upon two conditions: firstly, we verify that χ is a λ-list H-homomorphism, and secondly that $\forall_{e \in E_2(H)}\ |C(e)| = \alpha(e)$.

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 $\forall_{e \in E_2(H)}\ |C(e)| = \alpha(e)$. 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 $\forall_{e \in E_2(H)}\ |C(e)| = \alpha(e)$.

The claimed running time follows from the fact that $w(I') = \sum \alpha' = 3 \cdot \sum \alpha = O(d(I))$ and $|V(G')| = O(|E(G)|) = O(d(I) \cdot |V(G)|)$. ◀

4 An FPT-algorithm for List Allocation

In this section we give a brief description of the T-FPT-reductions required to prove that LA admits an FPT-algorithm, i.e., the proof of Theorem 3. This is the most technical part of our paper. Below we summarize the main steps.

1. List Allocation is T-FPT-reduced to its restriction, called CLA, where G is a connected graph and only $O(w)$ boxes are used. This reduction takes care of the different ways connected components of G can entirely be placed into the boxes and is based on dynamic programming.
2. CLA is T-FPT-reduced to a restriction of it, called HCLA, where G is highly connected in the sense that there is no set of w edges that can separate G into two "big" connected components. This reduction uses the technique of *recursive understanding*, introduced in [23] and further developed in [7] and [5] (see also [19]), for generalizations of the Multiway Cut problem).
3. HCLA is T-FPT-reduced to a special enhancement of it, called S-HCLA, 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 which are incident to crossing edges are contained in S. This variant of the problem permits the application of the technique of *randomized contractions*, introduced in [5].
4. Finally, S-HCLA is T-FPT-reduced to List Allocation restricted to instances whose sizes are bounded by a function of the parameter. It is a dynamic programming 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 $G \setminus S$.

5 Further research

In the definition of List Allocation we ask for a *λ-list H-homomorphism* of G where $\sum_{e \in E(H)} |C(e)| \leq \ell$. A different parameterization of List Allocation, that is similar in flavor to Min-Max Multiway Cut, may instead ask for a *λ-list H-homomorphism* of G where $\max_{v \in V(H)} \sum_{e \text{ is incident to } v} |C(e)| \leq \ell$. We call this new problem Max Bounded List Digraph Homomorphism (in short MBLDH) As it is straightforward to prove an analogue of Theorem 5, where BLDH is now replaced by MBLDH and instead of $2^{O(\ell \log h)} \cdot T(n,\ell)$ steps we now have a reduction that takes $2^{O(\ell^2 \log h)} \cdot T(n,\ell)$ steps. This implies that MBLDH, when parameterized by ℓ and h admits an FPT-algorithm that runs in $2^{O(\ell^2 \cdot \max\{\log \ell, \log h\})} \cdot n^4 \cdot \log n$ steps.

A natural research direction is to improve the running time of our FPT-algorithms for Min-Max Multiway Cut and Bounded List Digraph Homomorphism. If we want to improve our running times using the techniques used in this paper it seems that we need to crucially improve upon the recursive understanding and randomized contractions technique.

Acknowledgement. We would like to thank the anonymous referees of an earlier version of this paper for their thorough remarks and suggestions that improved the presentation and some proofs of the paper.

References

1 Nikhil Bansal, Uriel Feige, Robert Krauthgamer, Konstantin Makarychev, Viswanath Nagarajan, Joseph (Seffi) Naor, and Roy Schwartz. Min-max graph partitioning and small set expansion. In *Proc. of the 52nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 17–26. IEEE Computer Society, 2011.

2 Ronald Brown, Ifor Morris, J. Shrimpton, and Christopher D. Wensley. Graphs of morphisms of graphs. *Electronic Journal of Combinatorics*, 15(1), 2008.

3 Chandra Chekuri, Sudipto Guha, and Joseph Naor. The steiner k-cut problem. *SIAM Journal on Discrete Mathematics*, 20(1):261–271, 2006.

4 Rajesh Chitnis, MohammadTaghi Hajiaghayi, and Dániel Marx. Fixed-parameter tractability of directed multiway cut parameterized by the size of the cutset. In *Proc. of the 23rd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1713–1725. SIAM, 2012.

5 Rajesh Hemant Chitnis, Marek Cygan, MohammadTaghi Hajiaghayi, Marcin Pilipczuk, and Michal Pilipczuk. Designing FPT algorithms for cut problems using randomized contractions. In *Proc. of the 53rd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 460–469. IEEE Computer Society, 2012.

6 Rajesh Hemant Chitnis, László Egri, and Dániel Marx. List H-coloring a graph by removing few vertices. In *Proc. of the 21st Annual European Symposium on Algorithms (ESA)*, volume 8125 of *LNCS*, pages 313–324, 2013.

7 Marek Cygan, Daniel Lokshtanov, Marcin Pilipczuk, MichałPilipczuk, and Saket Saurabh. Minimum bisection is fixed parameter tractable. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing (STOC)*, pages 323–332. ACM, 2014.

8 E. Dahlhaus, D. S. Johnson, C. H. Papadimitriou, P. D. Seymour, and M. Yannakakis. The complexity of multiterminal cuts. *SIAM Journal on Computing*, (4):864–894, 1994.

9 Josep Díaz, Maria Serna, and Dimitrios M. Thilikos. Efficient algorithms for counting parameterized list H-colorings. *Journal of Computer and System Sciences*, 74(5):919–937, 2008.

10 Josep Díaz, Maria J. Serna, and Dimitrios M. Thilikos. (H, C, K)-coloring: Fast, easy, and hard cases. In *Proc. of the 26th International Symposium on Mathematical Foundations of Computer Science (MFCS)*, volume 2136 of *LNCS*, pages 304–315, 2001.

11 Josep Díaz, Maria J. Serna, and Dimitrios M. Thilikos. Fixed parameter algorithms for counting and deciding bounded restrictive list h-colorings. In *Proc. of the 12th Annual European Symposium on Algorithms (ESA)*, volume 3221 of *LNCS*, pages 275–286, 2004.

12 László Egri, Pavol Hell, Benoit Larose, and Arash Rafiey. Space complexity of list H-colouring: a dichotomy. In *Proc. of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 349–365. SIAM, 2014.

13 László Egri, Andrei Krokhin, Benoit Larose, and Pascal Tesson. The complexity of the list homomorphism problem for graphs. *Theory of Computing Systems*, 51(2):143–178, 2012.

14 Guy Even, Joseph Naor, Baruch Schieber, and Madhu Sudan. Approximating minimum feedback sets and multicuts in directed graphs. *Algorithmica*, 20(2):151–174, 1998.

15 Tomás Feder, Pavol Hell, and Jing Huang. List homomorphisms and circular arc graphs. *Combinatorica*, 19(4):487–505, 1999.

16 Tomás Feder, Pavol Hell, and Jing Huang. Bi-arc graphs and the complexity of list homomorphisms. *Journal of Graph Theory*, 42(1):61–80, 2003.

17 Naveen Garg, Vijay V. Vazirani, and Mihalis Yannakakis. Multiway cuts in node weighted graphs. *Journal of Algorithms*, 50(1):49–61, 2004.

18 Michel X. Goemans and David P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM*, 42(6):1115–1145, 1995.

19 Martin Grohe, Ken-ichi Kawarabayashi, Dániel Marx, and Paul Wollan. Finding topological subgraphs is fixed-parameter tractable. In *Proc. of the 43rd ACM Symposium on Theory of Computing (STOC)*, pages 479–488. ACM, 2011.

20 F. Hadlock. Finding a maximum cut of a planar graph in polynomial time. *SIAM Journal on Computing*, 4(3):221–225, 1975.

21 Pavol Hell and Jaroslav Nešetřil. *Graphs and homomorphisms*, volume 28 of *Oxford Lecture Series in Mathematics and its Applications.* Oxford University Press, Oxford, 2004.

22 Pavol Hell and Arash Rafiey. The dichotomy of list homomorphisms for digraphs. In *Proc. of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1703–1713. SIAM, 2011.

23 Ken ichi Kawarabayashi and Mikkel Thorup. The minimum k-way cut of bounded size is fixed-parameter tractable. In *Proc. of the 54th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 160–169. IEEE Computer Society, 2013.

24 David R. Karger, Philip Klein, Cliff Stein, Mikkel Thorup, and Neal E. Young. Rounding algorithms for a geometric embedding of minimum multiway cut. In *Proc. of the 31st Annual ACM Symposium on Theory of Computing (STOC)*, pages 668–678. ACM, 1999.

25 Lefteris M. Kirousis, Maria Serna, and Paul Spirakis. Parallel complexity of the connected subgraph problem. *SIAM Journal on Computing*, 22(3):573–586, 1993.

26 Dániel Marx. Parameterized graph separation problems. *Theoretical Computer Science*, 351(3):394–406, 2006.

27 Dániel Marx, Barry O'Sullivan, and Igor Razgon. Finding small separators in linear time via treewidth reduction. *ACM Transactions on Algorithms*, 9(4):30, 2013.

28 Dániel Marx and Igor Razgon. Fixed-parameter tractability of multicut parameterized by the size of the cutset. *SIAM Journal on Computing*, 43(2):355–388, 2014.

29 Harald Räcke. Optimal hierarchical decompositions for congestion minimization in networks. In *Proc. of the 40th Annual ACM Symposium on Theory of Computing (STOC)*, pages 255–264. ACM, 2008.

30 R. Ravi and Amitabh Sinha. Approximating k-cuts via network strength. In *Proc. of the 13th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 621–622. SIAM, 2002.

31 M. Stoer and F. Wagner. A simple min-cut algorithm. *Journal of the ACM*, 44(4):585–591, 1997.

32 Zoya Svitkina and Éva Tardos. Min-max multiway cut. In *Proc. of the 7th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX) and 8th International Workshop on Randomization and Computation (RANDOM)*, volume 3122 of *LNCS*, pages 207–218, 2004.