List H-coloring a graph by removing few vertices

Chitnis, Rajesh; Egri, László; Marx, Dániel

DOI:
10.1007/s00453-016-0139-6

License:
Other (please specify with Rights Statement)

Document Version
Peer reviewed version

Citation for published version (Harvard):
Chitnis, R, Egri, L & Marx, D 2017, 'List H-coloring a graph by removing few vertices', *Algorithmica*, vol. 78, no. 1, pp. 110-146. https://doi.org/10.1007/s00453-016-0139-6

List H-Coloring a Graph by Removing Few Vertices[⋆]

Rajesh Chitnis[1⋆⋆], László Egri[2⋆⋆⋆], and Dániel Marx[3]

[1] Weizmann Institute of Science, Rehovot, Israel, rajesh.chitnis@weizmann.ac.il
[2] Simon Fraser University, Burnaby, Canada, laszlo.egri@mail.mcgill.ca
[3] Institute for Computer Science and Control, Hungarian Academy of Sciences (MTA SZTAKI), Budapest, Hungary, dmarx@cs.bme.hu

Abstract. In the deletion version of the list homomorphism problem, we are given graphs G and H, a list $L(v) \subseteq V(H)$ for each vertex $v \in V(G)$, and an integer k. The task is to decide whether there exists a set $W \subseteq V(G)$ of size at most k such that there is a homomorphism from $G \setminus W$ to H respecting the lists. We show that DL-Hom(H), parameterized by k and $|H|$, is fixed-parameter tractable for any (P_6, C_6)-free bipartite graph H; already for this restricted class of graphs, the problem generalizes Vertex Cover, Odd Cycle Transversal, and Vertex Multiway Cut parameterized by the size of the cutset and the number of terminals. We conjecture that DL-Hom(H) is fixed-parameter tractable for the class of graphs H for which the list homomorphism problem (without deletions) is polynomial-time solvable; by a result of Feder et al. [11], a graph H belongs to this class precisely if it is a bipartite graph whose complement is a circular arc graph. We show that this conjecture is equivalent to the fixed-parameter tractability of a single fairly natural satisfiability problem, *Clause Deletion Chain-SAT*.

1 Introduction

Given two graphs G and H (without loops and parallel edges; unless otherwise stated, we consider only such graphs throughout this paper), a *homomorphism* $\phi : G \to H$ is a mapping $\phi : V(G) \to V(H)$ such that $\{u, v\} \in E(G)$ implies $\{\phi(u), \phi(v)\} \in E(H)$; the corresponding algorithmic problem *Graph Homomorphism* asks if G has a homomorphism to H. It is easy to see that G has a homomorphism into the clique K_c if and only if G is c-colorable; therefore, the algorithmic study of (variants of) Graph Homomorphism generalizes the study of graph coloring problems (cf. Hell and Nešetřil [18]). Instead of graphs, one can consider homomorphism problems in the more general context of relational structures. Feder and Vardi [14] observed that the standard framework for Constraint Satisfaction Problems (CSP) can be formulated as homomorphism problems for relational structures. Thus variants of Graph Homomorphism form a rich family of problems that are more general than classical graph coloring, but does not have the full generality of CSPs.

List Coloring is a generalization of ordinary graph coloring: for each vertex v, the input contains a list $L(v)$ of allowed colors associated to v, and the task is to find a coloring where each vertex gets a color from its list. In a similar way, *List Homomorphism* is a generalization of Graph Homomorphism: given two undirected graphs G, H and a list function $L : V(G) \to 2^{V(H)}$, the task is to decide if there exists a list homomorphism $\phi : G \to H$,

⋆ A preliminary version of the paper appeared in the proceedings of ESA 2013 [3]. Supported by ERC Starting Grant PARAMTIGHT (No. 280152) and OTKA grant NK105645.

⋆⋆ Most of this work was done when the author was a student at University of Maryland, College Park. Supported in part by NSF CAREER award 1053605, NSF grant CCF-1161626, ONR YIP award N000141110662, DARPA/AFOSR grant FA9550-12-1-0423, a University of Maryland Research and Scholarship Award (RASA) and a Summer International Research Fellowship from University of Maryland.

⋆⋆⋆ Supported by NSERC and FQRNT.

i.e., a homomorphism $\phi : G \to H$ such that for every $v \in V(G)$ we have $\phi(v) \in L(v)$. The List Homomorphism problem was introduced by Feder and Hell [10] and has been studied extensively [9, 13, 11, 12, 17, 20]. The problem of finding a list homomorphism from G to H is also referred to as List H-Coloring, since in the special case of $H = K_c$, the problem is equivalent to list coloring where every list is a subset of $\{1, \dots, c\}$.

An active line of research on homomorphism problems is to study the complexity of the problem when the target graph is fixed. Let H be an undirected graph. The Graph Homomorphism and List Homomorphism problems with fixed target H are denoted by $\textsc{Hom}(H)$ and L-$\textsc{Hom}(H)$, respectively. A classical result of Hell and Nešetřil [19] states that $\textsc{Hom}(H)$ is polynomial-time solvable if H is bipartite and NP-complete otherwise. For the more general List Homomorphism problem, Feder et al. [11] showed that L-$\textsc{Hom}(H)$ is in P if H is a bipartite graph whose complement is a circular arc graph, and it is NP-complete otherwise. Egri et al. [9] further refined this characterization and gave a complete classification of the complexity of L-$\textsc{Hom}(H)$: they showed that the problem is complete for NP, NL, or L, or otherwise the problem is first-order definable. In particular, they showed that L-$\textsc{Hom}(H)$ is in L if H is a (P_6, C_6)-free bipartite graph (that is, a bipartite graph that excludes the path P_6 on six vertices and the cycle C_6 on six vertices as induced subgraphs) and NL-hard otherwise.

In this paper, we increase the expressive power of (list) homomorphisms by allowing a bounded number of vertex deletions from the left-hand side graph G. Formally, in the DL-$\textsc{Hom}(H)$ problem we are given as input an undirected graph G, an integer k, a list function $L : V(G) \to 2^{V(H)}$ and the task is to output a *deletion set* $W \subseteq V(G)$ such that $|W| \leq k$ and the graph $G \setminus W$ has a list homomorphism to H, or an answer "no" if no such set exists. Let us note that DL-$\textsc{Hom}(H)$ is NP-hard already when H consists of a single isolated vertex: in this case the problem is equivalent to $\textsc{Vertex Cover}$, since removing the set W has to destroy every edge of G.

Our Results. We study the parameterized complexity of DL-$\textsc{Hom}(H)$ parameterized by the number of allowed vertex deletions and the size of the target graph H. Our goal is to characterize those graphs H for which DL-$\textsc{Hom}(H)$ is FPT. Clearly, if L-$\textsc{Hom}(H)$ is NP-complete, then DL-$\textsc{Hom}(H)$ is NP-complete already for $k = 0$, hence we cannot expect it to be FPT. Therefore, by results of Feder et al. [11], we need to consider only the case when H is a bipartite graph whose complement is a circular arc graph. We focus first on those graphs H for which the characterization of Egri et al. [9] showed that L-$\textsc{Hom}(H)$ is not only polynomial-time solvable, but also in logspace. As mentioned above, these graphs are precisely the (P_6, C_6)-free bipartite graphs, and in fact, they also admit a decomposition using certain simple operations (see Section 4 and [9]). To emphasize this decomposition, we also call this class of graphs *skew-decomposable graphs*. Note that the class of skew-decomposable graphs is a strict subclass of chordal bipartite graphs (P_6 is chordal bipartite but not skew-decomposable), and bipartite cographs and bipartite trivially perfect graphs are strict subclasses of skew-decomposable graphs.

The following examples show that even for simple skew-decomposable graphs H, DL-$\textsc{Hom}(H)$ is nontrivial, and in fact, we can express some well-studied problems this way:

– $\textsc{Vertex Cover}$ asks for a set of k vertices whose deletion removes every edge. This problem is equivalent to DL-$\textsc{Hom}(H)$ where H is a single vertex.

- Odd Cycle Transversal (also known as Vertex Bipartization) asks for a set of at most k vertices whose deletion makes the graph bipartite. This problem can be expressed by DL-Hom(H) when H consists of a single edge.
- In Vertex Multiway Cut parameterized by the size of the cutset and the number of terminals, a graph G is given with terminals $t_1, \dots, t_d$, and the task is to find a set of at most k vertices whose deletion disconnects t_i and t_j for any $i \neq j$. This problem can be expressed as DL-Hom(H) when H is a matching of d edges, in the following way. Let us obtain G' by subdividing each edge of G (making it bipartite) and let the list of t_i contain the vertices of the i-th edge e_i; all the other lists contain every vertex of H. It is easy to see that the deleted vertices must separate the terminals otherwise there is no homomorphism to H and, conversely, if the terminals are separated from each other, then the component of t_i has a list homomorphism to e_i.

Note that all three problems described above are NP-hard but known to be fixed-parameter tractable [6, 7, 25, 31].

Our first result is that the DL-Hom(H) problem is fixed-parameter tractable for the class of skew-decomposable graphs.

Theorem 1.1. DL-Hom(H) *is FPT parameterized by solution size and $|H|$, if H is restricted to be skew decomposable.*

That is, DL-Hom(H) can be solved in time $f(k, H) \cdot n^{O(1)}$ if H is skew decomposable, where f is a computable function that depends only of k and $|H|$ (see [7, 16, 29] for more background on fixed-parameter tractability).

As the graphs considered in the examples above are all skew-decomposable bipartite graphs, Theorem 1.1 is an algorithmic meta-theorem unifying the fixed-parameter tractability of Vertex Cover, Odd Cycle Transversal, and Vertex Multiway Cut parameterized by the size of the cutset and the number of terminals, and various combinations of these problems.

Theorem 1.1 shows that, for a particular class of graphs where L-Hom(H) is known to be polynomial-time solvable, the deletion version DL-Hom(H) is fixed-parameter tractable. We conjecture that this holds in general: whenever L-Hom(H) is polynomial-time solvable (i.e., the cases described by Feder et al. [11]), the corresponding DL-Hom(H) problem is FPT.

Conjecture 1.1. *If H is a fixed bipartite graph whose complement is a circular arc graph, then* DL-Hom(H) *is FPT parameterized by solution size.*

It might seem unsubstantiated to conjecture fixed-parameter tractability for every bipartite graph H whose complement is a circular arc graph, but we show that, in a technical sense, proving Conjecture 1.1 boils down to the fixed-parameter tractability of a single fairly natural problem. We introduce a variant of maximum ℓ-satisfiability, where the clauses of the formula are implication chains[4] $x_1 \to x_2 \to \cdots \to x_\ell$ of length at most ℓ, and the task is to make the formula satisfiable by removing at most k clauses; we call this problem *Clause Deletion ℓ-*Chain-SAT *(ℓ-CDCS)* (see Definition 5.1). We conjecture that for every fixed ℓ, this problem is FPT parameterized by k.

Conjecture 1.2. *For every fixed $\ell \geq 1$, Clause Deletion ℓ-*Chain-SAT *is FPT parameterized by solution size.*

[4] The notation $x_1 \to x_2 \to \cdots \to x_\ell$ is a shorthand for $(x_1 \to x_2) \wedge (x_2 \to x_3) \wedge \cdots \wedge (x_{\ell-1} \to x_\ell)$.

We show that for every bipartite graph H whose complement is a circular arc graph, the problem DL-Hom(H) can be reduced to ℓ-CDCS for some ℓ depending only on $|H|$. Somewhat more surprisingly, we are also able to show a converse statement: for every ℓ, there is a bipartite graph H_ℓ whose complement is a circular arc graph such that ℓ-CDCS can be reduced to DL-Hom(H_ℓ). That is, the two conjectures are equivalent. Therefore, in order to settle Conjecture 1.1, one necessarily needs to understand Conjecture 1.2 as well. Since the latter conjecture considers only a single problem (as opposed to an infinite family of problems parameterized by $|H|$), it is likely that connections with other satisfiability problems can be exploited, and therefore it seems that Conjecture 1.2 is a more promising target for future work.

Theorem 1.2. *Conjectures 1.1 and 1.2 are equivalent.*

Note that one may state Conjectures 1.1 and 1.2 in a stronger form by claiming fixed-parameter tractability with two parameters, considering $|H|$ and ℓ also as parameters (similarly to the statement of Theorem 1.1). One can show that the equivalence of Theorem 1.2 remains true with this version of the conjectures as well. However, stating the conjectures with fixed H and fixed ℓ gives somewhat simpler and more concrete problems to work on.

Even though we are unable to prove Conjecture 1.1, we can state a weaker result: a constant-factor approximation. Formally, let P be a parameterized problem where the parameter k is an integer appearing in the input and the task is to find some object of size at most k or report "no" if no such object exists (i.e., we are considering a minimization problem). Following [1, 26], we say that problem P is *fixed-parameter approximable (FPA) with ratio r* ($r \geq 1$) if there is an $f(k) \cdot n^{O(1)}$ time algorithm that either returns an object satisfying all output specifications except that its size is at most $r \cdot k$, or "no" and in the latter case it is guaranteed that there is no object of size at most k satisfying the output specifications.

Theorem 1.3. *If H is a fixed bipartite graph whose complement is a circular arc graph, then* DL-Hom(H) *is FPA with ratio $|H| + 1$, and the running time of the FPA-algorithm is $f(k, H) \cdot n^{O(1)}$.*

Note that we made no effort here to optimize the ratio $|H|+1$. We are stating Theorem 1.3 for two reasons. First, we get it essentially for free: it is easy to observe that ℓ-VDCS (a version of CDCS that is more convenient to work with) has an approximation algorithm with ratio ℓ by a reduction to the minimum cut problem, and hence the reduction from DL-Hom(H) to ℓ-VDCS appearing in Theorem 1.2 gives us a constant-factor approximation for DL-Hom(H) as well. Second, this approximation will be useful in the proof of Theorem 1.1, as it allows us to avoid the use of iterative compression in the induction step, which is crucial for obtaining an exponent independent of $|H|$.

Our Techniques: For our fixed-parameter tractability/approximability results, we use a combination of several techniques (some of them classical, some of them very recent) from the toolbox of parameterized complexity. Our first goal is to reduce DL-Hom(H) to the special case where for each vertex v, $L(v)$ contains vertices only from one or the other side of one component of the (bipartite) graph H; we call this special case the "fixed-side, fixed-component" version. We note that the reduction to this special case is non-trivial: as the examples above illustrate, expressing Odd Cycle Transversal seems to require that the lists contain vertices from both sides of H, and expressing Vertex Multiway Cut (parameterized by the size of the cutset and the number of terminals) seems to require that the lists contain vertices from more than one component of H. This suggests that a large part of the technical difficulty of the problem is encapsulated by this reduction.

We start our reduction by using the standard technique of iterative compression to obtain an instance where, besides a bounded number of precolored vertices, the graph is bipartite. We look for obvious conflicts in this instance. Roughly speaking, if there are two precolored vertices u and v in the same component of G with colors a and b, respectively, such that either (i) a and b are in different components of H, or (ii) a and b are in the same component of H but the parity of the distance between u and v is different from the parity of the distance between a and b, then the deletion set must contain a $u-v$ separator.

We use the treewidth reduction technique of Marx et al. [27] to obtain a bounded-treewidth region K of the graph that contains all such separators. As we know that K contains at least one deleted vertex, every component outside this region can contain at most $k-1$ deleted vertices. Thus we can recursively solve the problem for each such component, and collect this information in relations $R_1, \dots, R_m$, each relation having bounded arity. Finally, we are able to model the problem as a Monadic Second Order (MSO) formula (of a fixed size) over the graph of the region K, and the relations $R_1, \dots, R_m$, and evaluate this formula in linear time employing Courcelle's Theorem [5].

Even if the instance has no obvious conflicts as described above, we might still need to delete certain vertices due to more implicit conflicts. But now we know that for each vertex v, there is at most one component C of H and one side of C that is consistent with the precolored vertices appearing in the component of v, that is, the precolored vertices force this side of C on the vertex v. This seems to be close to our goal of being able to fix a component C of H and a side of C for each vertex. However, there is a subtle detail here: if the deleted set separates a vertex v from every precolored vertex, then the precolored vertices do not force any restriction on v. Therefore, it seems that at each vertex v, we have to be prepared for two possibilities: either v is reachable from the precolored vertices, or not. Unfortunately, this prevents us from assigning each vertex to one of the sides of a single component. We get around this problem by invoking the "randomized shadow removal" technique introduced by Marx and Razgon [28] (and subsequently used in [2, 4, 21, 22, 24]) to modify the instance in such a way that we can assume that the deletion set does not separate any vertex from the precolored vertices, hence we can fix the components and the sides.

By the chain of reductions described above, in order to prove Theorem 1.1 (FPT algorithm when H is skew decomposable), we need to solve the fixed-side, fixed-component version of the problem for skew-decomposable graphs. The inductive characterization of such graphs, given by [9] allows us to reduce the problem to subproblems with strictly simpler H, proving Theorem 1.1 inductively.

If H is a bipartite graph whose complement is a circular arc graph (recall that this class strictly contains all skew-decomposable graphs), then we show how to formulate the DL-Hom(H) problem as an instance of ℓ-CDCS (showing that Conjecture 1.2 implies Conjecture 1.1). Let us emphasize that our reduction to ℓ-CDCS works only if the lists of the DL-Hom(H) instance have the "fixed-side" property, and therefore our proof for the equivalence of the two conjectures (Theorem 1.2) utilizes the reduction machinery described above. For the reverse direction of Theorem 1.2, we give a self-contained proof with the construction of H_ℓ and the reduction from ℓ-CDCS to DL-Hom(H_ℓ).

Finally, to establish Theorem 1.3, we use parts of the above reduction machinery together with the aforementioned approximation algorithm for ℓ-VDCS.

It is interesting to point out the difference between the two algorithms establishing Theorems 1.1 and 1.3. In the algorithm for Theorem 1.3, once there are no conflicts in the sense explained above we reduce the problem to a number of minimum cut problems, and we are done. In the algorithm for Theorem 1.1, once there are no conflicts left we reduce the problem to a number of instances with smaller target graphs (and having some additional properties). However, now these new instances could again contain conflicts. We get rid of them as before, and repeat until we end up in one of the base cases: either the parameter k is reduced to 0, or the target graph becomes trivial.

2 Preliminaries

Given a graph G, let $V(G)$ denote its vertices and $E(G)$ denote its edges. If $G = (U, V, E)$ is bipartite, we call U and V the *sides or bipartite classes* of H. Let G be a graph and $W \subseteq V(G)$. Then $G[W]$ denotes the subgraph of G induced by the vertices in W. To simplify notation, we often write $G \setminus W$ instead of $G[V(G) \setminus W]$. The set $N(W)$ denotes the neighborhood of W in G, that is, the vertices of G which are not in W, but have a neighbor in W. Similarly to [27], we define two notions of separation: between two sets of vertices and between a pair (s,t) of vertices; note that in the latter case we assume that the separator is disjoint from s and t.

Definition 2.1. *A set S of vertices* separates *the sets of vertices A and B if no component of $G \setminus S$ contains vertices from both $A \setminus S$ and $B \setminus S$. If s and t are two distinct vertices of G, then an $s - t$ separator is a set S of vertices disjoint from $\{s,t\}$ such that s and t are in different components of $G \setminus S$.*

Definition 2.2. *Let G, H be graphs and L be a* list function $V(G) \to 2^{V(H)}$. *A list homomorphism ϕ from (G,L) to H (or if L is clear from the context, from G to H) is a homomorphism $\phi : G \to H$ such that $\phi(v) \in L(v)$ for every $v \in V(G)$. In other words, each vertex $v \in V(G)$ has a list $L(v)$ specifying the possible images of v. The right-hand side graph H is called the* target *graph.*

When the target graph H is fixed, we have the following problem:

L-Hom(H)
Input: A graph G and a list function $L : V(G) \to 2^{V(H)}$.
Question: Does there exist a list homomorphism from (G,L) to H?

The main problem we consider in this paper is the vertex deletion version of the L-Hom(H) problem, i.e., we ask if a set of vertices W can be deleted from G such that the remaining graph has a list homomorphism to H. Obviously, the list function is restricted to $V(G) \setminus W$, and for ease of notation, we denote this restricted list function $L|_{V(G)\setminus W}$ by $L \setminus W$. Since we will provide both an FPT and an FPA algorithm, we state our problems with the approximation ratio r, and note that setting $r = 1$ yields the FPT problem. We can now ask the following formal question:

DL-Hom(H) (with approximation ratio r)
Input: A graph G, a list function $L : V(G) \to 2^{V(H)}$, and an integer k.
Parameters: k , $|H|$
Output: A set $W \subseteq V(G)$ of size at most $r \cdot k$ such that there is a list homomorphism from $(G \setminus W, L \setminus W)$ to H, or "no", and then it is guaranteed that there is no such set of size at most k.

Notice that if $k = 0$ (and r is arbitrary), then DL-Hom(H) becomes L-Hom(H).

2.1 Iterative Compression

We finish the preliminaries with discussing the fairly standard iterative compression technique [31] adapted to our setting. We show that it is sufficient to solve the following compression problem:

DL-Hom(H) Compression (with approximation ratio r)
Input: A graph G_0, a list function $L : V(G_0) \to 2^{V(H)}$, an integer k, and a set $W_0 \subseteq V(G_0)$, $|W_0| \leq r \cdot k + 1$ such that $(G_0 \setminus W_0, L \setminus W_0)$ has a list homomorphism to H.
Parameters: $k, |H|$
Output: A set $W \subseteq V(G_0)$ disjoint from W_0 such that $|W| \leq r \cdot k$ and $(G_0 \setminus W, L \setminus W)$ has a list homomorphism to H, or "no", and then it is guaranteed that there is no such set of size at most k.

In order to be able to prove the approximation result Theorem 1.3, we defined the compression problem in a way that it may return a constant-factor approximation of the optimum. However, this does not create any complication in the application of iterative compression.

Lemma 2.3. *Assume that there is an FPA-algorithm with approximation ratio r for the problem* DL-Hom(H) Compression *whose running time is $f(k)\cdot n^c$, where f is some function depending only on k, and c is a constant. Then there is an FPA-algorithm with approximation ratio r for the problem* DL-Hom(H) *whose running time is $g(k) \cdot n^{c+1}$, where g is a function depending only on k.*

Proof. Assume that $V(G) = \{v_1, \ldots, v_n\}$ and for $i \in [n]$, let $V_0 = \emptyset$ and $V_i = \{v_1, \ldots, v_i\}$. We construct a sequence of subsets $X_0 \subseteq V_0, X_1 \subseteq V_1, \ldots, X_n \subseteq V_n$ such that X_i is a deletion set of size at most $r \cdot k$ for the instance $(G[V_i], L|_{V_i})$ of DL-Hom(H) (i.e., deleting X_i from $G[V_i]$ allows for a homomorphism to H). Trivially, $X_0 = \emptyset$ is a deletion set for $(G[V_0], L|_{V_0})$.

Observe that if X_i is a deletion set of size at most $r \cdot k$ for $(G[V_i], L|_{V_i})$, then $X_i \cup \{v_{i+1}\}$ is a deletion set for $(G[V_{i+1}], L|_{V_{i+1}})$ of size at most $r \cdot k + 1$. Therefore, for each $i \in [n-1]$, we set $W_0 = X_i \cup \{v_{i+1}\}$ and use, as a black-box, an algorithm for DL-Hom(H) Compression to construct a deletion set X_{i+1} of size at most $r \cdot k$ for $(G[V_{i+1}], L|_{V_{i+1}})$. There is a technical difficulty here: the specification of DL-Hom(H) Compression asks for a deletion set W *disjoint* from W_0. However, there is an easy and standard way of resolving this problem: we invoke the algorithm for DL-Hom(H) Compression $2^{|W_0|} = 2^{O(rk)}$ times, trying all possible intersection of W_0 and the solution W we are looking for.

If the algorithm for DL-Hom(H) Compression returns that there is no deletion set of size at most k for $(G[V_i], L|_{V_i})$ for some $i \in [n]$, then there is no such deletion set for the whole graph G. Therefore, in this case the algorithm can return a "no" answer. Moreover, since $V_n = V(G)$, if all the calls to the compression algorithm are successful, then X_n is a deletion set of size at most $r \cdot k$ for the graph G. □

We note that the compression algorithm needs an actual solution as part of its input, and therefore in all reduction steps, we will carry around with us a solution. We note that producing an actual solution for our subproblems will occasionally be non-trivial.

3 Reducing the Bipartite Compression Problem

At this stage, we have to solve the compression problem. We start with a few observations that allow us a reduction to the so-called *bipartite compression* problem. Consider the compression problem from Section 2.1. Since the new solution W is required to be disjoint from W_0, every list homomorphism from $(G_0 \setminus W, L \setminus W)$ induces a partial list homomorphism from $(G_0[W_0], L|_{W_0})$ to H. We guess all such partial list homomorphisms γ from $(G_0[W_0], L|_{W_0})$ to H, and we hope that we can find a set W disjoint from W_0 such that γ can be extended to a total list homomorphism from $(G_0 \setminus W, L \setminus W)$ to H. To guess these partial homomorphisms, we simply enumerate all possible mappings from W_0 to H and check whether the given mapping is a list homomorphism from $(G_0[W_0], L|_{W_0})$ to H. If not, we discard the given mapping. Observe that we need to consider only $|V(H)|^{|W_0|} \leq |V(H)|^{r \cdot k+1}$ mappings. Hence, in what follows we can assume that we are given a partial list homomorphism γ from $G_0[W_0]$ to H.

We propagate the consequences of γ to the lists of the vertices in the neighborhood of W_0, as follows. For each neighbor $u \in N(W_0)$ of $v \in W_0$, we trim $L(u)$ as $L(u) \leftarrow L(u) \cap N(\gamma(v))$. Since H is bipartite, the list of each vertex in $N(W_0)$ is now a subset of one of the sides of a single connected component of H. We say that such a list assignment is *fixed side* and *fixed component*.

Recall that $G_0 \setminus W_0$ has a list homomorphism ϕ to the bipartite graph H, and therefore $G_0 \setminus W_0$ must be bipartite. However, after propagating the effect of γ to the neighborhood of W_0 as above, the homomorphism ϕ may not be a valid list homomorphism for $G_0 \setminus W_0$. Therefore, we keep only the restriction of ϕ to $G_0 \setminus (W_0 \cup N(W_0))$, which we denote by ϕ_0. To summarize the properties of the problem we have at hand, we define it formally below. Note that we do not need the graph G_0 and the set W_0 any more, only the graph $G_0 \setminus W_0$, and the neighborhood $N(W_0)$. To simplify notation, we refer to $G_0 \setminus W_0$ and $N(W_0)$ as G and N_0, respectively.

DL-Hom(H) Bipartite-Compression (with approximation ratio r), denoted by BC(H)
Input: A bipartite graph G, a list function $L : V(G) \to 2^{V(H)}$, a set $N_0 \subseteq V(G)$, where for each $v \in N_0$, the list $L(v)$ is fixed side and fixed component, and an integer k. Furthermore, it is assumed that there is a list homomorphism ϕ_0 from $G \setminus N_0$ to H.
Parameters: k, $|H|$
Output: A set $W \subseteq V(G)$, such that $|W| \leq r \cdot k$ and $(G \setminus W, L \setminus W)$ has a list homomorphism to H, or "no", and then it is guaranteed that there is no such set of size at most k.

The main goal of this section is to further reduce the problem to the *fixed-side, fixed-component* problem. To define this problem, consider a generic instance (G, L, N_0, k) of

BC(H). Let $u, v \in V(G)$, assume that $\{u, v\} \in E(G)$, and that $L(u)$ and $L(v)$ are fixed-side, fixed-component H-lists. We say that $L(u)$ and $L(v)$ are *pairwise consistent*, if $L(u)$ and $L(v)$ are subsets of the same component C of H, and of different sides of C. If for all such pairs u and v the lists $L(u)$ and $L(v)$ are pairwise consistent, then we say that the (fixed-side, fixed-component) list function L is *consistent (with respect to H)*. The formal definition of the problem is given below.

DL-Hom(H) Fixed-Side Fixed-Component (with approximation ratio r), denoted by **FS-FC(H)**
Input: A bipartite graph G, a consistent fixed-side, fixed-component list function $L : V(G) \to 2^{V(H)}$, and an integer k.
Parameters: k, $|H|$
Output: A set $W \subseteq V(G)$ such that $|W| \leq r \cdot k$ and $G \setminus W$ has a list homomorphism to H, or "no", and then it is guaranteed that there is no such set of size at most k.

The main technical result that we prove in the following section is that BC(H) can be reduced to DL-Hom(H) Fixed-Side Fixed-Component.

Theorem 3.1. *If the* FS-FC(H) *problem is FPA (where H is bipartite) with approximation ratio r and running time $f(k, H) \cdot x^c \cdot \log^d x$ for some function f depending only on k and H (where x is the size of the input, i.e., the number of vertices in the input graph, and c is a sufficiently large constant), then the* BC(H) *problem is also FPA with approximation ratio r and running time $g(k, H) \cdot x^c \cdot \log^{d+1} x$ for some function g depending only on k and H.*

Note again that in the special case when $r = 1$, the above theorem can be understood as an FPT reduction result.

Theorems 1.1 and 1.3 are proved through Theorem 3.1. To prove Theorem 1.3, we first use iterative compression (see Section 2.1) to reduce to BC(H) and then Theorem 3.1 to reduce to FS-FC(H). If H is a bipartite graph whose complement is a circular arc graph, then we are able to obtain a polynomial-time approximation algorithm for FS-FC(H) with ratio $|V(H)| + 1$. Note that this approximation ratio is propagated through the reduction steps, eventually becoming the ratio for the final approximation algorithm. In fact, for this approximation result we need only the "fixed-side" property of the lists, the "fixed-component" property is not required; we denote by FS(H) this generalization of FS-FC(H). For reference, let us state this approximation result formally (the proof appears in Section 5).

Corollary 5.8. *Let H be a bipartite graph whose complement is a circular arc graph. Then there is a polynomial-time approximation algorithm for FS(H) with ratio $|V(H)| + 1$.*

For the proof of Theorem 1.1, we use double induction: roughly, after reducing the problem to the bipartite compression version of the problem, we assume that there is an FPT algorithm $\mathcal{A}$ for the bipartite compression problem for every parameter value strictly smaller than k and for every proper subgraph of H. To solve an instance of FS-FC(H) when H is skew decomposable, we observe that the inductive construction of skew-decomposable graphs allows us to reduce the problem to instances where H has strictly smaller size. Then we can invoke $\mathcal{A}$ inductively with strictly smaller H. Additionally, if some obvious conflicts arise, we need to branch on deleting one endpoint of an edge and this time, we can invoke $\mathcal{A}$ with smaller parameter value.

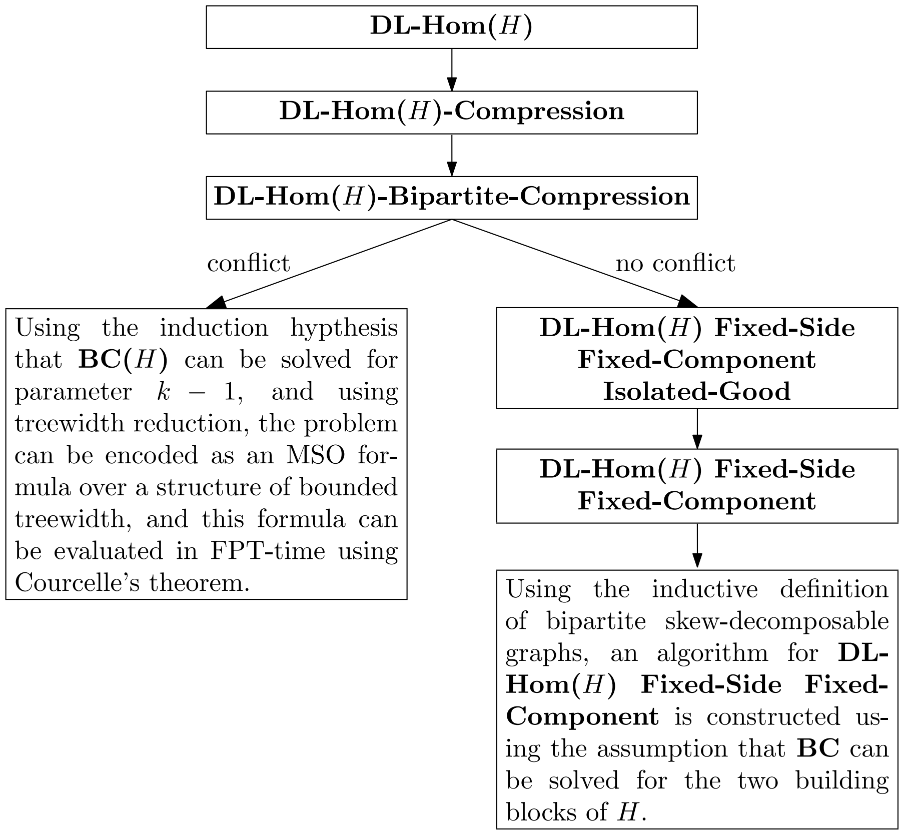
DL-Hom(H)
DL-Hom(H)-Compression
DL-Hom(H)-Bipartite-Compression
conflict
no conflict
Using the induction hypthesis that **BC(H)** can be solved for parameter $k - 1$, and using treewidth reduction, the problem can be encoded as an MSO formula over a structure of bounded treewidth, and this formula can be evaluated in FPT-time using Courcelle's theorem.
DL-Hom(H) Fixed-Side Fixed-Component Isolated-Good
DL-Hom(H) Fixed-Side Fixed-Component
Using the inductive definition of bipartite skew-decomposable graphs, an algorithm for **DL-Hom(H) Fixed-Side Fixed-Component** is constructed using the assumption that **BC** can be solved for the two building blocks of H.

Fig. 1. The structure of the reductions that establish the fixed-parameter tractability of DL-Hom(H) when H is a skew-decomposable graph.

To facilitate understanding, it is a good place to present the main road map of the two algorithms in Figure 1. We started with iterative compression to reduce the problem to BC(H), and the main task is to show how BC(H) can be solved recursively. Given an instance of BC(H), we proceed the following way. We look for certain "conflicts" between vertices of N_0 (formally defined in the next section), which imply that two vertices of N_0 have to be separated in the solution. If we find such a conflict, then we follow the conflict branch, where we use the treewidth reduction technique of Marx et al. [27] to reduce the problem to several instances with strictly smaller parameter and we jump back to bipartite compression. Although the number m of branches is not bounded by a function of k and $|H|$, the overall running time can still be shown to be sufficiently small. Intuitively, this is because the recursion calls BC(H) for subinstances that partition the original input. The base case (when no vertices are allowed to be removed) can be solved using a known algorithm for the list-homomorphism problem (see, e.g., [11]).

If there is no conflict, we follow the "no conflict" branch in the figure. Here, the main technical result is to reduce the FS-FC-IG(H) problem to the FS-FC(H) problem. We note that the branching factor here will include $\log(|V(G)|)$, coming from the "shadow removal" technique of Marx and Razgon [28]. Note that this is the source of the log factors in Theorem 3.1.

3.1 The case when there is a conflict

We define two types of *conflicts* between the vertices of N_0. Recall that the lists of the vertices in N_0 in a $\mathrm{BC}(H)$ instance are fixed side and fixed component.

Definition 3.2. *Let (G, L, N_0, k) be an instance of* $\mathrm{BC}(H)$*. Let u and v be vertices of N_0 in the same component of G. We say that u and v are in* component conflict *if $L(u)$ and $L(v)$ are subsets of vertices of different components of H. Furthermore, u and v are in* parity conflict *if u and v are not in component conflict, and either*

- *u and v belong to the same side of G but $L(u)$ is a subset of one of the sides of a component of C of H and $L(v)$ is a subset of the other side of C,*
- *or u and v belong to different sides of G but $L(u)$ and $L(v)$ are subsets of the same side of a component of H.*

The following lemma is immediate from the definitions.

Lemma 3.3. *Let (G, L, N_0, k) be an instance of* $\mathrm{BC}(H)$*. If u and v are any two vertices in N_0 that are in component or parity conflict, then any solution W must contain a set S that separates the sets $\{u\}$ and $\{v\}$.*

In this section, we handle the case when a conflict, as described in Definition 3.2 exists, and the other case is handled in Section 3.2. If a conflict exists, its presence allows us to invoke the treewidth reduction technique of Marx et al. [27] to split the instance into a bounded-treewidth part, and into instances having parameter value strictly less than k. After solving these instances with smaller parameter value recursively, we encode the problem in Monadic Second Order (MSO) logic, and apply Courcelle's theorem [5]. In fact, we will apply a version of this result, stated by Flum et al. [15] that can be used to output a solution for the problem.

Before we can prove the main lemma of this section (Lemma 3.10), first we need the definitions of tree decomposition and treewidth.

Definition 3.4. *A* tree decomposition *of a graph G is a pair $(T, \mathcal{B})$ in which T is a tree and $\mathcal{B} = \{B_i \mid i \in V(T)\}$ is a family of subsets of $V(G)$ such that*

1. *$\bigcup_{i \in V(T)} B_i = V(G)$;*
2. *For each $e \in E(G)$, there exists an $i \in V(T)$ such that $e \subseteq B_i$;*
3. *For every $v \in V(G)$, the set of nodes $\{i \in I \mid v \in B_i\}$ forms a connected subtree of T.*

The width *of a tree decomposition is the number* $\max\{|B_t| - 1 \mid t \in V(T)\}$*. The* treewidth $\mathrm{tw}(G)$ *is the minimum of the widths of the tree decompositions of G.*

It is well known that the maximum clique size of a graph is at most its treewidth plus one.

A *vocabulary* τ is a finite set of relation symbols or predicates. Every relation symbol R in τ has an arity associated to it. A relational structure $\mathbf{A}$ over a vocabulary τ consists of a set A, called the domain of $\mathbf{A}$, and a relation $R^{\mathbf{A}} \subseteq A^r$ for each $R \in \tau$, where r is the arity of R.

Definition 3.5. *The Gaifman graph of a τ-structure $\mathbf{A}$ is the graph $G_{\mathbf{A}}$ such that $V(G_{\mathbf{A}}) = A$ and $\{a, b\}$ $(a \neq b)$ is an edge of $G_{\mathbf{A}}$ if there exists an $R \in \tau$ and a tuple $(a_1, \ldots, a_r) \in R^{\mathbf{A}}$ such that $a, b \in \{a_1, \ldots, a_r\}$, where r is the arity of R. The treewidth of $\mathbf{A}$ is defined as the treewidth of the Gaifman graph of $\mathbf{A}$.*

The result we need from [27] states that all the minimal $s-t$ separators of size at most k in G can be covered by a set C inducing a bounded-treewidth subgraph of G. In fact, a stronger statement is true: this subgraph has bounded treewidth even if we introduce additional edges in order to take into account connectivity outside C. This is expressed by the operation of taking the torso:

Definition 3.6. *Let G be a graph and $C \subseteq V(G)$. The graph* $\text{torso}(G, C)$ *has vertex set C and two vertices $a, b \in C$ are adjacent if $\{a, b\} \in E(G)$ or there is a path in G connecting a and b whose internal vertices are not in C.*

Observe that by definition, $G[C]$ is a subgraph of $\text{torso}(G, C)$.

Lemma 3.7 ([27]). *Let s and t be two vertices of G. For some $k \geq 0$, let C_k be the union of all minimal sets of size at most k that are $s-t$ separators. There is an $O(g_1(k)\cdot(|E(G)|+|V(G)|))$ time algorithm that returns a set $C \supseteq C_k \cup \{s, t\}$ such that the treewidth of* $\text{torso}(G, C)$ *is at most $g_2(k)$, for some functions g_1 and g_2 of k.*

Lemma 3.3 gives us a pair of vertices that must be separated. Lemma 3.7 specifies a bounded-treewidth region C of the input graph which must contain at least one vertex of the above separator, that is, we know that at least one vertex must be deleted in this bounded-treewidth region.

Courcelle's Theorem gives an easy way of showing that certain problems are linear-time solvable on bounded-treewidth graphs: it states that if a problem can be formulated in MSO[5], then there is a linear-time algorithm for it. This theorem also holds for relational structures of bounded treewidth instead of just graphs, a generalization we need because we introduce new relations $R_1, \ldots, R_m$ to encode the properties of the components of $G \setminus C$. We note that the arity of these relations will be bounded by a function of k, resulting from the fact that any component of $G \setminus C$ has a bounded (in k) number of neighbors in $G[C]$.

Theorem 3.8 (Courcelle's Theorem, cf. [16]). *The following problem is fixed-parameter tractable:*

$\mathbf{p}^* - \mathbf{tw} - \text{MC(MSO)}$
Input : A structure $\mathbf{A}$ and an MSO-sentence φ;
Parameter : $\text{tw}(\mathbf{A}) + \lvert\varphi\rvert$;
Problem : Decide whether $\mathbf{A} \models \varphi$.

Moreover, there is a computable function f and an algorithm that solves it in time $f(|\varphi|, \text{tw}(\mathbf{A}))\cdot |A| + O(|\mathbf{A}|)$.

The more general result, which also returns a solution, can be stated the following way:

Theorem 3.9 (Flum et al. [15, Corollary 4.15]). *There exist a function $f : N \times N \to N$ and an algorithm that, given a structure $\mathbf{A}$ and an MSO-formula $\varphi(X_1, \ldots, X_l, x_1, \ldots, x_m)$ decides in time $f(|\varphi|, tw(\mathbf{A})) \cdot |A|$ if there are $B_1, \ldots, B_l \subseteq A$ and $a_1, ..., a_m \in A$ such that $\mathbf{A} \models \varphi(B_1, \ldots, B_l, a_1, \ldots, a_m)$, and, if this is the case, computes such sets and elements.*

[5] For background on MSO, we refer the reader to, e.g., the textbook of Flum and Grohe [16]; very briefly, MSO is a logical language that allows quantification over the elements and subsets of the universe of a relational structure.

In the lemma below, we first show how to solve the decision version of the bipartite compression problem. This is not enough though, one reason being that in the recursive call, we actually assume that we have a deletion set we can work with. At the end of Lemma 3.10, we explain how we can keep track of information and use Theorem 3.9 to output an actual (approximate) solution.

Lemma 3.10. *Let $\mathcal{A}$ be an FPA algorithm with approximation ratio r that correctly solves $\mathrm{BC}(H)$ for input instances in which the first parameter is at most $k-1$. Suppose that the running time of $\mathcal{A}$ is $f(k-1, H) \cdot x^c \cdot \log^d x$, where x is the size of the input (the number of vertices in the input graph), c is a sufficiently large constant, and d is some non-negative integer. Let I be an instance of $\mathrm{BC}(H)$ with parameter k that contains a component or parity conflict. Then for instance I, either a solution of size at most $r \cdot k$ can be produced, or a "no"answer, in which case it is guaranteed that no solution of size at most k exists. The running time of the algorithm is $f(k, H) \cdot x^c \cdot \log^d x$ (where f is inductively defined at the end of the proof).*

Proof. For clarity, first we solve the decision version of the problem with approximation ratio $r = 1$, and explain after how to handle the general case. Let $I = (G, L, N_0, k)$ be an instance of $\mathrm{BC}(H)$. Let $v, w \in N_0$ such that v and w are in component or parity conflict. Then by Lemma 3.3, the deletion set must contain a $v-w$ separator. Using Lemma 3.7, we can find a set C with the properties stated in the lemma (and note that we will also make use of the functions g_1 and g_2 in the statement of the lemma). Most importantly, C contains at least one vertex that must be removed in any solution, so the maximum number of vertices that can be removed from any connected component of $G[V(G) \setminus C]$ without exceeding the budget k is at most $k-1$. Therefore, the outline of our strategy is the following. We use $\mathcal{A}$ to solve the problem for some slightly modified versions of the components of $G[V(G) \setminus C]$, and using these solutions, we construct an MSO formula that encodes our original problem I. Furthermore, since the relational structure over which this MSO formula must be evaluated has bounded treewidth, and the size of the formula depends only on $|H|$ and k (in particular, it is independent of the size of the input structure), the formula can be evaluated in linear time using Theorem 3.8.

Assume without loss of generality that $V(H) = \{1, \ldots, h\}$. The MSO formula has the form

$$\exists K_0, \ldots, K_h \Big[\varphi_{\mathrm{part}}(K_0, \ldots, K_h) \wedge \varphi_C(K_0, \ldots, K_h) \wedge \bigvee_{i=0}^{k} \left(\varphi_{|K_0| \leq i}(K_0) \wedge \varphi_{\bar{C}, k-i}(K_0, \ldots, K_h) \right) \Big].$$

The set K_0 represents the deletion set that is removed from $G[C]$, and $K_1, \ldots, K_h$ specifies the colors of the vertices in the subgraph $G[C \setminus K_0]$. The sub-formula $\varphi_{\mathrm{part}}(K_0, \ldots, K_h)$ checks if $K_0, \ldots, K_h$ is a valid partition of C, and φ_C checks if $K_1, \ldots, K_h$ is an H-coloring of $G[C \setminus K_0]$. The third subformula checks whether there is an additional set $X \subseteq V(G) \setminus C$ such that $|K_0| + |X| \leq k$, and the coloring $K_1, \ldots, K_h$ of $G[C \setminus K_0]$ can be extended to $G[V(G) \setminus (K_0 \cup X)]$. In this part, the formula $\varphi_{|K_0| \leq i}(K_0)$ checks if the size of K_0 is at most i, and the formula $\varphi_{\bar{C}, k-i}(K_0, \ldots, K_h)$ checks if the coloring of $G[C \setminus K_0]$ can be extended with $k-i$ additional deletions. Thus the disjunction is true if the set X with $|K_0| + |X| \leq k$ exists.

In what follows, we describe how to construct these subformulas, and we also construct the relational structure $\mathbf{S}$ from G over which this formula must be evaluated. To simplify the presentation, we refer to $K_0, \ldots, K_h$ as a coloring, even if the vertices in K_0 are not mapped to $V(H)$ but removed.

The formula $\varphi_{\mathbf{part}}$. To check whether $K_0, \ldots, K_h$ is a partition of $V(G)$, we use the formula

$$\varphi_{\text{part}} \equiv \left(\forall x \bigvee_{i=0}^{h} K_i(x)\right) \wedge \left(\forall x \bigwedge_{i\neq j} \neg(K_i(x) \wedge K_j(x))\right).$$

The formula φ_C. To check whether a partition $K_0, \ldots, K_h$ is a list homomorphism from G to H, we first define unary relations $U_t, t \in \{1, \ldots, h\}$, such that $U_t(v)$ holds if and only if $t \in L(v)$. Note that adding a unary relation to $\mathbf{S}$ does not increase its treewidth. The following formula checks if $K_0, \ldots, K_h$ is a list-homomorphism.

$$\begin{aligned}\varphi_C(K_0, \ldots, K_h) \equiv \\ \forall x, y \left((\neg K_0(x) \wedge \neg K_0(y) \wedge E(x,y)) \rightarrow \left(\bigvee_{(i,j)\in E(H)} (K_i(x) \wedge K_j(y))\right)\right) \wedge \\ \bigwedge_{i=1}^{h} \left(\forall x\ (K_i(x) \rightarrow U_i(x))\right).\end{aligned}$$

The formula $\varphi_{|K_0|\leq j}$. To check whether $|K_0| \leq j$, we use the formula

$$\varphi_{|K_0|\leq j} \equiv \neg \exists x_1, \ldots, x_{j+1} \left(\bigwedge_{i=1}^{j+1} K_0(x_i) \wedge \bigwedge_{i\neq i'\ 1\leq i,i'\leq j+1} (x_i \neq x_{i'})\right).$$

The formula $\varphi_{\bar{C},j}$. First we construct a set of "indicator" predicates. For all $q \in \{1, \ldots, g_2(k)+1\}$ (where g_2 is from Lemma 3.7), for all q-tuples $(c_1, \ldots, c_q) \in \{0, 1, \ldots, h\}^q$, and for all $\ell \in \{0, \ldots, j\}$, we produce a predicate $R = R_{(c_1,\ldots,c_q),\ell}$ of arity q. Intuitively, the meaning of a tuple $(v_1, \ldots, v_q)$ being in this relation is that if $\{v_1, \ldots, v_q\}$ is a clique in the torso and has the coloring $(c_1, \ldots, c_q)$ (where $c_i = 0$ means that the vertex is deleted), then to extend this coloring to the components of $G \setminus C$ that attach precisely to the clique $\{v_1, \ldots, v_q\}$, at least ℓ further deletions in these components are required. Formally, we place a q-tuple $(v_1, \ldots, v_q) \in V(G)^q$ into R using the procedure below. (We argue later how to do this in FPT-time.)

Fix an arbitrary ordering $\prec$ on the vertices of C. The purpose of $\prec$ will be to avoid counting the number of vertices that must be removed from a single component more than once, as we will see later. Let D be the union of all components of $G[V(G) \setminus C]$ whose neighborhood in C is precisely $\{v_1, \ldots, v_q\}$, and assume without loss of generality that $v_1 \prec \cdots \prec v_q$. We call such a union of components a *common neighborhood component*. For each such D, for each $i \in [q]$, if $c_i \neq 0$, then for all neighbors u of v_i in D, remove any vertex of H from the list $L(u)$ which is not a neighbor of c_i. Let L' be the new lists obtained this way. Observe that the coloring $(c_1, \ldots, c_q)$ of the vertices $(v_1, \ldots, v_q)$ can be extended to (D, L) after removing j vertices from D if and only if (D, L') can be H-colored after removing j vertices from D. Now we use algorithm $\mathcal{A}$ (details given later) to determine the minimum number z of such deletions. The tuple $(v_1, \dots, v_q)$ is placed into R if $z \geq \ell$ $(\star)$ (see the discussion in the last paragraph for the case when $r > 1$). Observe that if we did not order $\{v_1, \dots, v_q\}$ according to $\prec$, then $\{v_1, \dots, v_q\}$ would be associated with more than one indicator relation, which would lead to counting the vertices needed to be removed from D multiple times.

Let $R_1, \dots, R_m$ be an enumeration of all possible $R_{(c_1,\dots,c_q),\ell}$ as defined above. Let $\mathbf{S}$ be the relational structure $(C; E(G[C]), R_1, \dots, R_m)$. Observe that if $(v_1, \dots, v_q)$ is a tuple in one of these relations, then $\{v_1, \dots, v_q\}$ is a clique in $\operatorname{torso}(G, C)$, since it is the neighborhood of a component of $G \setminus C$. Thus the Gaifman graph of $\mathbf{S}$ is a subgraph of $\operatorname{torso}(G, C)$, which means that $\operatorname{tw}(\mathbf{S}) \leq g_2(k)$. Moreover, for every component of $G \setminus C$, as its neighborhood in C is a clique in $\operatorname{torso}(G, C)$, the neighborhood cannot be larger than $g_2(k) + 1$: a graph with treewidth at most $g_2(k)$ has no clique larger than $g_2(k) + 1$.

We express the statement that a coloring of $G[C]$ *cannot* be extended to $G \setminus C$ with at most j deletions by stating that there is a subset of (common neighborhood) components of $G \setminus C$ such that the total number of deletions needed for these components is more than j. We construct a separate formula for each possible way the required number of deletions can add up to more than j and for each possible coloring appearing on the neighborhood of these components. Formally, we define a formula ψ for every combination of

- integer $0 \leq t \leq j$ (number of common neighborhood components considered),
- integers $1 \leq q_1, \dots, q_t \leq g_2(k) + 1$ (sizes of the neighborhoods of components),
- integers $c^i_1, \dots, c^i_{q_i}$ for every $1 \leq i \leq t$ (colorings of the neighborhoods), and
- integers $0 \leq \ell_1, \ell_2, \dots, \ell_t \leq j + 1$ such that $\sum_{i=1}^{t} \ell_i \geq j + 1$ (number of deletions required in the neighborhoods)

in the following way:

$$\begin{aligned}\psi(K_0, \dots, K_h) \equiv \exists x_{1,1}, \dots, x_{1,q_1}, x_{2,1}, \dots, x_{2,q_2}, \dots, x_{t,1}, \dots, x_{t,q_t} \\ \left(\bigwedge_{i \neq j, 1 \leq i,j \leq t} (q_i = q_j) \to (x_{i,1} \neq x_{j,1} \vee \dots \vee x_{i,q_i} \neq x_{j,q_j}) \right) \wedge \\ \bigwedge_{i=1}^{t} \Big(K_{c^i_1}(x_{i,1}) \wedge \dots \wedge K_{c^i_{q_i}}(x_{i,q_i}) \wedge R_{(c^i_1,\dots,c^i_{q_i}),\ell_i}(x_{i,1}, \dots, x_{i,q_i}) \Big).\end{aligned}$$

Let $\psi_1, \dots, \psi_p$ be an enumeration of all these formulas. (Notice that the size and the number of these formulas is bounded by a function of k.) We define

$$\varphi_{\bar{C},j}(K_0, \dots, K_h) \equiv \neg \bigvee_{i=1}^{p} \psi_i.$$

We argue now that $\varphi_{\bar{C},j}$ is true if and only if it suffices to remove j additional vertices. It follows from the definition that given an H-coloring $K_0, \dots, K_h$ of $G[C]$, if $\varphi_{\bar{C},j}$ is false, then there is a subset of the components $G \setminus C$ witnessing that at least $j + 1$ vertices must be removed from $G[V(G) \setminus C]$ in order to extend the coloring $K_0, \dots, K_h$ to $G \setminus C$.

Conversely, assume that more than j vertices must be removed from $G[V(G) \setminus C]$ in order to extend the coloring $K_0, \dots, K_h$. Then there are neighborhoods $N_1, \dots, N_t \subseteq C$ with $t \leq j + 1$ such that at least $j + 1$ vertices must be removed from the components of $G[V(G) \setminus C]$ whose neighborhoods are among $N_1, \dots, N_t$. By definition, this is detected by one of the ψ_i's in the disjunction, and therefore $\varphi_{\bar{C},j}$ is false.

Running time of the decision algorithm and the inputs for $\mathcal{A}$. For the running time, by the comments above and by Theorem 3.8, we just need to give an upper bound on the time to construct the relations $R_1, \dots, R_m$. First we need to determine the common neighborhood components. Let $D_1, \dots, D_p$ be the components of $G[V(G) \setminus C]$. Find $N(D_1)$ (note that $N(D_1) \subseteq C$), and find all other components in the list $D_1, \dots, D_p$ having the same neighborhood (in C) as D_1. This produces the common neighborhood component of D_1. To find the next common neighborhood component, find the smallest j such that $N(D_j) \neq N(D_1)$, and find all other components among $D_1, \dots, D_p$ that have the same neighborhood (in C) as D_j. This produces the common neighborhood component of D_j. We repeat this procedure until all common neighborhood components are determined. Let $Q_1, \dots, Q_n$ be an enumeration of all the common neighborhood components.

Observe that $V(Q_i) \cap V(Q_j) = \emptyset$ whenever $i \neq j$, implying $\sum_{i=1}^{n} |V(Q_i)| \leq |V(G)|$. For each Q_i, for all possible colorings of $N(Q_i)$, all possible ways of removing at most k vertices from $N(Q_i)$ (which is at most $\binom{g_2(k)+1}{k}$), we determine the lists L' as described above. Then we run $\mathcal{A}$ on (Q_i, L') with parameters $0, 1, \dots, k-1$ to determine the smallest number of vertices that must be removed. To view (Q_i, L') as an instance of $\mathrm{BC}(H)$, observe that Q_i is bipartite. We need to specify a set N_0^i for (Q_i, L'), which is just $V(Q_i) \cap N_0$. The property that $L(v)$ is fixed side and fixed component is obviously inherited, and clearly, there is a list homomorphism from $Q_i \setminus N_0^i$ to H.

Resuming the main argument, assume that $N(Q_i) = \{v_1, \dots, v_q\}$, where $v_1 \prec \dots \prec v_q$. Then if $(c_1, \dots, c_q)$ is the tuple that encodes the current vertex coloring and the vertices removed from $N(Q_i)$, and at least ℓ vertices have to be removed from Q_i, then $(v_1, \dots, v_q)$ is placed into the relation $R_{(c_1,\dots,c_q),\ell}$.

The number of times we run $\mathcal{A}$ for Q_i (for different modifications L' of the lists of the vertices of Q_i) is $h(k, H)$ for some h depending only on k and $|H|$, and $|N(Q_i)| \leq g_2(k) + 1$. Recall that the running time of $\mathcal{A}$ is $f(k-1, H) \cdot x^c \cdot \log^d x$, where x is the size of the input. Therefore, the total time $\mathcal{A}$ is running is at most

$$\sum_{i=1}^{n} h(k,H) \cdot f(k-1,H) \cdot |V(Q_i)|^c \log^d |V(Q_i)| \leq$$
$$h(k,H) \cdot f(k-1,H) \cdot \left(\sum_{i=1}^{n} |V(Q_i)| \right)^c \log^d \left(\sum_{i=1}^{n} |V(Q_i)| \right) \leq$$
$$h(k,H) \cdot f(k-1,H) \cdot |V(G)|^c \log^d |V(G)|.$$

The first inequality follows from the convexity of the function $x^c \cdot \log^d(x)$ when $x \geq 1$ $(c, d \geq 0)$.

Returning a solution and handling the case when $r > 1$. Now we argue how we can output $K_0 \cup X$, which is a solution with the desired properties. By Theorem 3.9, we can produce a collection of sets $\{K_0, K_1, \dots, K_h\}$ that make our MSO formula true in FPT-time. To output X, we keep track of the deletion sets produced by $\mathcal{A}$ when run on a tuple $(v_1, \dots, v_q)$ that was considered to be placed into a relation $R_{(c_1,\dots,c_q),\ell}$. To do this, we color and remove vertices from C as indicated by the sets $K_0, K_1, \dots, K_h$. Then for each common neighborhood component Q_i, we check which vertices of $N(Q_i)$ are removed (in K_0), and which are colored with what color. From this information, we can construct L' as above, and once we have this, we can look up the deletion set returned by $\mathcal{A}$ on input (Q_i, L').

When $r > 1$, the MSO formula for the problem is the same as before. Note that in this case at $(\star)$ above, the tuple $(v_1, \ldots, v_q)$ is placed into R if $\mathcal{A}$ outputs "no" when run with parameter ℓ, in which case, the properties of $\mathcal{A}$ guarantee that there is no solution of size at most ℓ. If the MSO formula is satisfiable, then we can use K_0 together with the r-approximate solutions for the subproblems to obtain an r-approximate solution for the overall problem. Conversely, if there is a solution of size at most k for the overall problem, then we can use this solution to construct a satisfying assignment for the MSO formula. $\square$

3.2 The case when there is no conflict

Recall that, in the BC(H) problem, the lists of the vertices in N_0 are fixed side and fixed component, and by assumption, there exists a list homomorphism ϕ_0 from $G \setminus N_0$ to H. To ensure that the list function L is consistent fixed side and fixed component, we process the BC(H) instance in the following way. First, if a component of G does not contain any vertex of N_0, then this component can be colored using ϕ_0. Hence such components can be removed from the instance without changing the problem. Consider a component C of G and let v be a vertex in $C \cap N_0$. Recall that $L(v)$ is fixed side and fixed component by the definition of BC(H); let H_v be the component of H such that $L(v) \subseteq H_v$ in H, and let $(S_v, \bar{S}_v)$ be the bipartition of H_v such that $L(v) \subseteq S_v$. For every vertex u in C that is in the same side of C as v, let $L'(u) = L(u) \cap S_v$; for every vertex u that is in the other side of C, let $L'(u) = L(u) \cap \bar{S}_v$. Note that since the instance does not contain any component or parity conflicts, this operation on u is the same no matter which vertex $v \in C \cap N_0$ is selected: every vertex in $C \cap N_0$ forces $L(u)$ to the same side of the same component of H. The definition of L' is motivated by the observation that if u remains connected to v in $G \setminus W$, then u has to use a color from $L'(u)$: its color has to be in the same component H_v as the colors in $L(v)$, and whether it uses colors from S_v or $\bar{S}_v$ is determined by whether it is on the same side as $L(v)$ or not.

If the consistent fixed-side fixed-component instance (G, L', N_0, k) has a solution, then clearly (G, L, N_0, k) has a solution as well. Unfortunately, the converse is not true: by moving to the more restricted set L', we may lose solutions. The problem is that even if a vertex u is in the same side of the same component of G as some $v \in N_0$, if u is separated from v in $G \setminus W$, then the color of u does not have to be in the same side of the same component of H as $L(v)$; therefore, restricting $L(u)$ to $L'(u)$ is not justified. However, we observe that the vertices of G that are separated from N_0 in $G \setminus W$ do not significantly affect the solution: if C is a component of $G \setminus W$ disjoint from N_0, then ϕ_0 can be used to color C. Therefore, we redefine the problem in a way that if a component of $G \setminus W$ is disjoint from N_0, then it is "good" in the sense that we do not require a coloring for these components.

DL-Hom(H) Fixed-Side Fixed-Component Isolated-Good (with approximation ratio r), denoted by FS-FC-IG(H)
Input: A bipartite graph G, a consistent fixed-side fixed-component list function $L : V(G) \to 2^{V(H)}$, a set of vertices $N_0 \subseteq V(G)$, and an integer k.
Parameters: k, $|H|$
Output: A set $W \subseteq V(G)$ such that $|W| \le r \cdot k$ and for every component C of $G \setminus W$ with $C \cap N_0 \neq \emptyset$, there is a list homomorphism from $(G[C], L|_C)$ to H, or "no", and then it is guaranteed that there is no such set of size at most k.

If the instance (G, L, N_0, k) of BC(H) has a solution, then the modified FS-FC-IG(H) instance (G, L', N_0, k) also has a solution: for every component C of $G \setminus W$ intersecting N_0, the vertices in $C \cap N_0$ force every vertex of C to respect the more restricted lists L'. Conversely, a solution of instance (G, L', N_0, k) of FS-FC-IG(H) can be turned into a solution for instance (G, L, N_0, k) of BC(H): for every component of $G \setminus W$ intersecting N_0, the coloring using the lists L' is a valid coloring also for the less restricted lists L and each component disjoint from N_0 can be colored using ϕ_0. Thus we have established a reduction from BC(H) to FS-FC-IG(H). In the rest of this section, we further reduce FS-FC-IG(H) to FS-FC(H), thus completing the proof of Theorem 3.1.

3.3 Reducing FS-FC-IG(H) to FS-FC(H)

If we could ensure that the solution W has the property that $G \setminus W$ has no component C disjoint from N_0, then FS-FC-IG(H) and FS-FC(H) would be equivalent. Intuitively, we would like to somehow remove every such component C from the instance to ensure this equivalence. This seems to be very difficult for at least two reasons: first, we do not know the deletion set W (finding it is what the problem is about), hence we do not know where these components are, and, second, it is not clear how to argue that removing certain sets of vertices does not make the problem easier. Nevertheless, the "shadow removal" technique of Marx and Razgon [28] does precisely this: it allows us to remove components separated from N_0 in the solution.

Let us explain how the shadow removal technique can be invoked in our context. We need the following definitions:

Definition 3.11. (closest) *Let $S \subseteq V(G)$. We say that a set $R \supseteq S$ is an S-closest set if there is no $R' \subset R$ with $S \subseteq R'$ and $|N(R')| \leq |N(R)|$.*

Definition 3.12. (reach) *Let G be a graph and $A, X \subseteq V(G)$. Then $R_{G \setminus X}(A)$ is the set of vertices reachable from a vertex in $A \setminus X$ in the graph $G \setminus X$.*

The following lemma connects these definitions with our problem: we argue that solving FS-FC-IG(H) essentially requires finding a closest set. For technical reasons, we construct a new auxiliary graph G' from G by adding a new vertex s to G, and all edges of the form $\{s, v\}$, $v \in N_0$. We fix a deletion set W^* for the FS-FC-IG(H)-instance G which we will use throughout the rest of this section:

> Among all deletion sets for G that have minimum size, W^* is chosen so that $R_{G' \setminus W^*}(\{s\}) = R_{G \setminus W^*}(N_0) \cup \{s\}$ has minimum size. (1)

We also set $R^* = R_{G' \setminus W^*}(\{s\})$.

Lemma 3.13. *It holds that $W^* = N(R^*)$ and R^* is an $\{s\}$-closest set.*

Proof. We note that $s \notin W^*$. Clearly, $N(R^*) \subseteq W^*$. If $W^* \neq N(R^*)$, then let us define $W' = N(R^*)$. Now $G \setminus W^*$ and $G \setminus W'$ have the same components intersecting N_0: every vertex of $W^* \setminus W'$ is in a component of $G \setminus W'$ that is disjoint from N_0. Therefore, FS-FC-IG(H) has a solution with deletion set W', contradicting the minimality of W^*.

If R^* is not an $\{s\}$-closest set, then there exists a set R' such that $\{s\} \subseteq R' \subset R^*$ and $|N(R')| \leq |N(R^*)| = |W^*|$. Let $W' = N(R')$, we have $|W'| \leq |W^*| \leq k$. We now claim that W'

can be used as a deletion set for FS-FC-IG(H). If we show this, then $R_{G'\setminus W'}(\{s\}) \subseteq R' \subset R^*$ contradicts the minimality of W^*.

For a vertex x, let $C_G(x)$ denote the vertices of the component of G that contains x. We now show that if $x \in N_0$, then $C_{G\setminus W'}(x) \subseteq C_{G\setminus W^*}(x)$. Let $x \in N_0$ and $y \in C_{G\setminus W'}(x)$. Then x, y are in the same component of R', and hence also in R^* as $R' \subset R^*$, i.e., $y \in C_{G\setminus W^*}(x)$ and therefore $C_{G\setminus W'}(x) \subseteq C_{G\setminus W^*}(x)$ indeed holds. This shows that W' is also a solution, since we know that W^* is a solution for FS-FC-IG(H), i.e., each component of $G \setminus W^*$ which intersects N_0 has a homomorphism to H, and hence so does any subgraph. ☐

The following theorem states the derandomized version of the shadow removal technique.

Theorem 3.14 (Marx and Razgon [28, Theorem 3.17]). *There is an algorithm* DeterministicSets(G, S, k) *that, given an undirected graph G, a set $S \subseteq V(G)$, and an integer k, produces $t = 2^{O(k^3)} \cdot \log |V(G)|$ subsets $Z_1, Z_2, \ldots, Z_t$ of $V(G) \setminus S$ such that the following holds: For every S-closest set R with $|N(R)| \leq k$, there is at least one $i \in [t]$ such that*

1. *$N(R) \cap Z_i = \emptyset$, and*
2. *$V(G) \setminus (R \cup N(R)) \subseteq Z_i$.*

The running time of the algorithm is $2^{O(k^3)} \cdot n^{O(1)}$.

By Lemma 3.13 we know that $R^* = R_{G'\setminus W^*}(\{s\})$ is an $\{s\}$-closest set and $W^* = N(R^*)$. Thus we can use Theorem 3.14 with $S = \{s\}$ to construct the sets $Z_1, \ldots, Z_t$. Then we branch on choosing one such $Z = Z_i$, and we can assume for the rest of this section that we have a set Z satisfying the following properties:

$$W^* \cap Z = \emptyset \quad \text{and} \quad V(G) \setminus (R^* \cup W^*) \subseteq Z. \tag{2}$$

That is, Z does not contain any vertex of the deletion set W^*, but it completely covers the set of vertices separated from N_0 by W^*, and possibly covers some other vertices not separated from N_0. Now we show how to use this property of the set Z to reduce FS-FC-IG(H) to FS-FC(H).

For each component C of $G[Z]$, we run the decision algorithm (see, for example, [11]) for L-Hom(H) with the list function $L|_C$. If C has no list homomorphism to H, then we call C a *bad component* of Z; otherwise, we call C a *good component* of Z. The following lemma shows that all neighbors of a bad component C in the graph $G \setminus Z$ must be in the solution W^*.

Lemma 3.15. *Let Z be a set satisfying* (2) *above. If C is a bad component of $G[Z]$ (i.e., $(C, L|_C)$ has no list homomorphism to H), then all vertices of the neighborhood of C in $G \setminus Z$ belong to W^*.*

Proof. Recall that by assumption, Z contains any vertex that is separated from N_0 by W^*. Therefore, if a neighbor v of C is in $G \setminus Z$, then v is connected to N_0 in $G \setminus W^*$. It follows that C is also connected to N_0 as Z (and hence C) is disjoint from W^*. Since $(C, L|_C)$ has no list homomorphism to H, this contradicts that W^* is a solution for FS-FC-IG(H). ☐

By Lemma 3.15, we may safely remove the neighborhood of every bad component C (decreasing the parameter k appropriately) and then, as the component C becomes separated from N_0, we can remove C as well. More precisely, we define

$$B_Z = \{v \mid v \text{ is a vertex in a bad component}\}$$

and

$$X_Z = \{v \mid v \notin Z \text{ and } v \text{ has a neighbor in } B_Z\}.$$

The following lemma concludes our reduction.

Lemma 3.16. *If the FS-FC-IG(H)-instance $I_1 = (G, L, N_0, k)$ has a deletion set, then there is a set Z among the sets $Z_1, \dots, Z_t$ returned by the algorithm in Theorem 3.14 (on input G', $S = \{s\}$, and parameter k) such that the following holds:*

(a) The FS-FC(H)-instance $I_2 = (G \setminus (X_Z \cup B_Z), L|_{G\setminus(X_Z\cup B_Z)}, k - |X_Z|)$ has a deletion set.
(b) If W'' is an r-approximate deletion set for I_2, then $W'' \cup X_Z$ is an r-approximate deletion set for I_1.

Moreover, given Z, we can compute I_2 from I_1 and $W'' \cup X_Z$ from W'' in polynomial time.

Proof. Assume that FS-FC-IG(H) has a deletion set, and choose this set to be W^*, as defined at (1). Also choose Z to be the set defined at (2).

To prove (a), let φ be a list homomorphism from the components of $G \setminus W^*$ that contain a vertex of N_0 to H. We set $W' = W^* \setminus X_Z$, and claim that W' is a solution for I_2. By Lemma 3.15, we have that X_Z is a subset of W^*. Therefore the size of W' is clearly at most $k - |X_Z|$, and $(G \setminus (X_Z \cup B_Z)) \setminus W' = (G \setminus (X_Z \cup B_Z)) \setminus (W^* \setminus X_Z)) = (G \setminus W^*) \setminus B_Z$. Thus, it is sufficient to show that there is a list homomorphism from $(G \setminus W^*) \setminus B_Z$ to H. Recall that Z contains all components separated from N_0 by W^*, and for each such component we checked whether there was a list homomorphism to H. The (bad) components which did not have a list homomorphism to H are not present in $(G \setminus W^*) \setminus B_Z$. For the (good) components which had a list homomorphism ψ to H, we can obviously just use ψ. Since the rest of the components have a vertex from N_0, for these components we can use φ.

For (b), suppose that the FS-FC(H)-instance I_2 has an r-approximate deletion set W''. Then the set $W'' \cup X_Z$ is an r-approximate deletion set for the instance (G, L, N_0, k) of FS-FC-IG(H): every vertex of B_Z is separated from N_0 by X_Z, and W'' is a solution for the rest of G. Observe that if $|W''| \le r \cdot (k - |X_Z|)$, then $|W'' \cup X_Z| = |W''| + |X_Z| \le r \cdot (k - |X_Z|) + |X_Z| \le r \cdot k$, since $r \ge 1$.

Finally, polynomial-time computability of I_2 and $W'' \cup X_Z$ is clear from the comments above. □

Note that this concludes the proof of Theorem 3.1:

Proof (of Theorem 3.1). An instance I of BC(H) is given. We induct on k. Assume first that I contains a component or parity conflict. Then the algorithm for BC(H) described in Lemma 3.10 reduces I to a number of smaller BC(H) instances. This recursion proceeds as long as the smaller instances contain a conflict. Note that as shown in Lemma 3.10, the approximation ratios are preserved. If some branch reaches $k = 0$, then we can solve the instance using any known polynomial-time decision algorithm for the problem (see, e.g., [11]).

If I contains no conflict, then I is reduced to an instance of FS-FC-IG (see the discussion in Section 3.2). This FS-FC-IG instance is reduced to a number of FS-FC instances, one for each Z_i obtained from Theorem 3.14. If the FS-FC-IG instance has a deletion set, then there is at least one set Z among the Z_i-s that guarantees properties (a) and (b) in Lemma 3.16. As shown in that lemma, the approximation ratios are preserved.

For the running time analysis, we also induct on k. If the BC(H) instance contains a component or parity conflict, then we obtain the claim directly from Lemma 3.10. If there is no conflict, then BC(H) is reduced to $2^{O(k^3)} \cdot \log |V(G)|$ FS-FC instances (see the branching at Theorem 3.14). ☐

4 Solving the BC(H) problem for skew-decomposable graphs

We prove Theorem 1.1 in this section. Since we need an FPT algorithm, in all the reductions above, we set $r = 1$. Essentially, the principal mechanism behind the FPT-algorithm is a double induction on k and the inductive construction (defined below) of skew-decomposable graphs. The induction on k is already taken care of by Theorem 3.1, so we need to worry only about the induction on the construction of H. Roughly speaking, we solve an instance of FS-FC(H) by solving $f(k, H)$ instances of BC(H') for some proper subgraphs H' of H. Since H' is a proper subgraph of H, the depth of the recursion is at most $|H|$. Using Theorem 3.1, at every iteration, we obtain an additional $\log x$ factor in the running time, and therefore the overall running time at the end will be at most $f'(k, H) \cdot x^c \cdot \log^{|H|}(x)$ (where f' is some function of k and H). This can be bounded by $f'(k, H) \cdot x^{c+1}$: it is well known that $\log^k n$ can be bounded, for example, by $2^{O(k^2)} \cdot n$.

We define the inductive construction of skew-decomposable graphs below. We note that this construction was used to inductively build a logspace algorithm for L-Hom(H), where H is skew decomposable (Egri et al. [9]), and in a somewhat similar manner, we also use this construction here to build our FPT algorithm. The *special sum* operation is an operation to compose bipartite graphs.

Definition 4.1. ***(special sum)*** *Let H_1, H_2 be two bipartite graphs with bipartite classes T_1, B_1 and T_2, B_2, respectively, such that neither T_1 nor B_2 is empty. The* special sum $H_1 \oslash H_2$ *is obtained by taking the disjoint union of the graphs, and adding all edges $\{u, v\}$ such that $u \in T_1$ and $v \in B_2$.*

Definition 4.2. ***(skew decomposable)*** *A bipartite graph H is called* skew decomposable *if $H \in \mathcal{S}$, where the graph class $\mathcal{S}$ is defined as follows:*

- *$\mathcal{S}$ contains the graph that is a single vertex;*
- *If $H_1, H_2 \in \mathcal{S}$, then their disjoint union $H_1 \uplus H_2$ also belongs to $\mathcal{S}$;*
- *If $H_1, H_2 \in \mathcal{S}$, then $H_1 \oslash H_2$ also belongs to $\mathcal{S}$.*

The following two lemmas, together with the base case of the induction discussed after the lemmas, complete the proof.

Lemma 4.3. *Assume that $H = H_1 \uplus H_2$. (Note that H could have more than 2 connected components.) Then we can produce a solution for the* FS-FC(H) *instance G with at most $f(k, H)$ calls to BC(H_1) or BC(H_2), where the inputs have size at most the size of G, and the parameters are at most k.*

Proof. Since G is component consistent, each connected component of G has lists that are all subsets of either $V(H_1)$ or $V(H_2)$. Let G_1 be the subgraph of G that is induced by all those vertices whose lists are subsets of $V(H_1)$. We similarly define G_2. For $i = 1, 2$, let k_i be the smallest integer such that there is a set $W^i \subseteq V(G_i)$ of size at most k_i such that there is a list homomorphism from $G_i \setminus W^i$ to H_i. Then clearly, there is a deletion set for G if and only if $k_1 + k_2 \leq k$. Note that if there are such k_1 and k_2, then the solution set for G is just the union of the solutions for G_1 and G_2. We focus on G_1 (and proceed similarly for G_2). We try to find a solution with parameters $k_1 = 0, \ldots, k$, so let's fix a specific $k_1 \leq k$. By the definition of FS-FC(H), the bipartite graph G_1 has consistent fixed-side fixed-component H_1-lists. Thus, we can use Corollary 5.8 (whose proof appears in Section 5) to find an approximate solution W_0^1 of size at most $s = k(|V(H_1)| + 1)$ for the FS(H_1)-problem instance G_1. This is sufficient for our purposes, as we show now. Following the same steps as in the iterative compression section in the Preliminaries (except that instead of W_0, we use W_0^1 as our starting solution) and the beginning of Section 3, we can reduce this problem to solving $g(k, H)$ instances of BC(H_1). □

Lemma 4.4. *Assume that $H = H_1 \oslash H_2$. Then we can produce a solution for the* FS-FC(H) *instance G with at most $f(k, H)$ calls to BC(H_1) or BC(H_2), where the inputs have size at most the size of G, and the parameters are at most k.*

Proof. Let (G, L) be an instance of FS-FC(H). Assume that the bipartite classes of H_i are T_i, B_i, $i \in \{1, 2\}$ and we have added all edges between T_1 and B_2 in the special sum operation. For any $u \in V(G)$ such that $L(u) \subseteq T_1 \cup T_2$ and $L(u) \cap T_1 \neq \emptyset$, we take the intersection of $L(u)$ and T_1 to obtain $L^\star(u)$ (we "trim" the lists). Similarly, for every $v \in V(G)$ such that $L(v) \subseteq B_1 \cup B_2$ and $L(v) \cap B_2 \neq \emptyset$, we obtain $L^\star(v) \leftarrow L(v) \cap B_2$. Because for any $x_1 \in T_1$ and any $x_2 \in T_2$ it holds that $N(x_1) \supseteq N(x_2)$, and for any $y_1 \in B_1$ and any $y_2 \in B_2$ it holds that $N(y_2) \supseteq N(y_1)$, it is easy to see that (G, L) and $(G, L^\star)$ are equivalent.

If $\{u, v\}$ is an edge such that $L^\star(u) \subseteq B_1$ and $L^\star(v) \subseteq T_2$, we call $\{u, v\}$ a *bad edge*: clearly, we must remove at least one endpoint of a bad edge. We branch on which endpoint of a bad edge to remove (from both G and $L^\star$) until either there are no more bad edges, or we exceed the budget k, in which case we abort the current computation branch. Note that we produce at most 2^k instances without bad edges. From now on we can assume that there are no bad edges.

Recall that there is a bipartite clique on T_1 and B_2. This has the consequence that if $\{u, v\}$ is an edge of G such that $L^\star(u) \subseteq T_1$ and $L^\star(v) \subseteq B_2$, then no matter to which element of $L^\star(u)$ the vertex u is mapped, and no matter to which element of $L^\star(v)$ the vertex v is mapped, the edge $\{u, v\}$ is always mapped to an edge of H. Therefore, we can simply remove these edges from G without changing the solution space. Let $G^\star$ be this modified version of G. Observe now that for any connected component C of $G^\star$, for every edge $\{u, v\} \in E(C)$ (where u is in the top bipartite class of G), we have that either $L^\star(u) \subseteq T_1$ and $L^\star(v) \subseteq B_1$, or $L^\star(u) \subseteq T_2$ and $L^\star(v) \subseteq B_2$. That is, no edge can be mapped to the edges between T_1 and B_2. It follows that C has lists that are all subsets of either $V(H_1)$ or $V(H_2)$ (we note that C is not necessarily component consistent, because H_1 or H_2 could consist of more than one component). Since $G^\star$ meets the condition in the first sentence of Lemma 4.3 (i.e., each connected component of $G^\star$ has lists that are all subsets of either $V(H_1)$ or $V(H_2)$), now we can proceed exacty the same way as we did there. □

The base case, when H consists of a single vertex, is simply the fixed-parameter tractable vertex cover problem. In fact, since the input is bipartite, we can alternatively use König's theorem to solve the base case optimally. To sum up, we proved Theorem 1.1.

Theorem 1.1. DL-Hom(H) *is FPT parameterized by solution size and* $|H|$, *if* H *is restricted to be skew decomposable.*

5 Relation between DL-Hom(H) and Satisfiability Problems

The purpose of this section is to prove Theorem 1.2: the equivalence of DL-Hom(H) with the Clause Deletion ℓ-Chain SAT (ℓ-CDCS) problem (defined below), in the cases when L-Hom(H) is characterized as polynomial-time solvable by Feder et al. [11], that is, when H is a bipartite graph whose complement is a circular arc graph. This satisfiability problem belongs to the family of clause deletion problems (e.g., Almost 2-SAT [30, 6, 23]), where the goal is to make a formula satisfiable by the deletion of at most k clauses. At the end of the section, the reductions proven here will allow us to finish the construction of the FPA-algorithm.

Definition 5.1. *A* chain clause *is a conjunction of the form*

$$(x_0 \to x_1) \wedge (x_1 \to x_2) \wedge \cdots \wedge (x_{m-1} \to x_m),$$

where x_i and x_j are different variables if $i \neq j$. The length *of a chain clause is the number of variables it contains. (A chain clause of length* 1 *is a variable, and it is satisfied by both possible assignments.) To simplify notation, we denote chain clauses of the above form as*

$$x_0 \to x_1 \to \cdots \to x_m.$$

An ℓ-Chain-SAT *formula consists of:*

- *a set of variables V;*
- *a set of chain clauses over V such that any chain clause has length at most ℓ;*
- *a set of unary clauses (a unary clause is a variable or its negation).*

Clause Deletion ℓ-Chain-SAT (ℓ-CDCS)
Input: An ℓ-Chain-SAT formula F.
Parameter: k
Output: A set of clauses of size at most k such that removing these clauses from F makes F satisfiable, or "no" if no such set exists.

5.1 The variable-deletion version

For technical reasons, it will be convenient to work with a variant of the problem where instead of constraints, certain sets of variables are allowed to be removed, a certain disjointness condition is required, and some chain clauses of length 2 behave differently from chain clauses having length 1 or length at least 3:

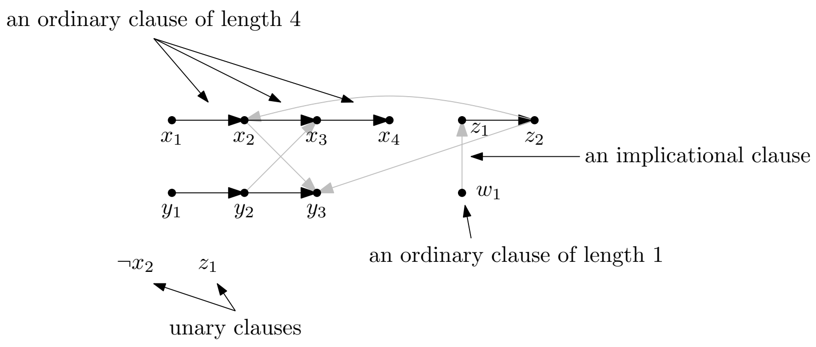

Fig. 2. Illustration of a VDCS instance. Chains of black arrows represent ordinary clauses. Gray arrows represent implicational clauses. Unary clauses are given explicitly as $\neg x_2$ and z_1.

Variable Deletion ℓ-Chain-SAT (ℓ-VDCS)
Input: An ℓ-Chain-SAT formula F (containing some unary and chain clauses) in which the chain clauses are partitioned into *ordinary* and *implicational* clauses satisfying the following properties. The ordinary clauses are on pairwise vertex-disjoint sets of variables, and each implicational clause is of length exactly 2, with the two variables appearing in two distinct ordinary clauses.
Parameter: k
Output: A set of at most k ordinary clauses such that removing all variables of these ordinary clauses, and also removing any clause that contains any of these variables, makes F satisfiable, or "no" if no such set exists.

The following two lemmas show the equivalence of the two versions of the problem. Note that the first reduction increases the value of ℓ, but the equivalence holds in the sense that ℓ-CDCS is FPT for every fixed ℓ if and only if ℓ-VDCS is FPT for every fixed ℓ.

To simplify the exposition of the proofs that follow, we introduce some terminology. In the context of the CDCS problem, we say that a clause is *undeletable* if it has at least $k + 1$ identical copies in the given formula, where k is the maximum number of clauses allowed to be removed. By "adding an undeletable clause" we mean adding $k + 1$ copies of the given clause, and by "making a clause undeletable" we mean adding sufficiently many copies of that clause so that it becomes undeletable.

Lemma 5.2. *There is a parameterized reduction from ℓ-VDCS to $(2\ell + 2)$-CDCS.*

Proof. Let F be a given ℓ-VDCS instance. Before we construct a CDCS instance with parameter k, we transform F into a more standard form. First we obtain an instance F_1 from F as follows. Let $x_0 \to \cdots \to x_m$ be an ordinary clause of F with the property that there exist indices $i \leq j$ such that F contains the unary clauses x_i and $\neg x_j$. The variables of such a clause must be removed in any solution, so we remove all variables $x_0, \ldots, x_m$ from F (and any clause that contains any of these variables), and reduce the parameter accordingly. Clearly, F is equivalent to F_1 (with the reduced parameter).

We produce an instance F_2 from F_1 as follows. Let $C = x_0 \to \cdots \to x_m$ be an ordinary clause such that there are indices $i < j$ such that F_1 contains unary clauses $\neg x_i$ and x_j. Take the largest index i such that F_1 contains the unary clause $\neg x_i$, and the smallest index j such that F_1 contains the unary clause x_j. Observe that in any satisfying variable assignment of this clause, for any $i' < i$, the variable $x_{i'}$ must take on the value 0. Therefore, we collapse all variables $x_{i'}$, $i' \leq i$, into the variable x_i. Using a similar reasoning, we can collapse all variables $x_{j'}$, $j \leq j'$ into x_j. Notice that F_2 can be made satisfiable by k deletions if and only if F_1 can be made satisfiable by k deletions.

By construction, if $x_0 \to x_1 \to \cdots \to x_m$ is an ordinary clause of F_2, then if this clause contains a negative unary clause, it must be $\neg x_0$, and if it contains a positive unary clause, it must be x_m. Furthermore, if both unary clauses $\neg x_0$ and x_m are present, then $m \geq 1$.

We are now ready to construct the $(2\ell+2)$-CDCS instance F' from F_2. For each ordinary clause $x_0 \to x_1 \to \cdots \to x_m$ of F_2, we place a chain clause

$$x_0 \to x_0' \to \tilde{x}_0 \to x_1 \to x_1' \to x_2 \to x_2' \to \cdots \to x_{m-1} \to x_{m-1}' \to \tilde{x}_m \to x_m \to x_m'$$

into F'. If $\neg x_0$ is a unary clause in F_2, then we add the unary clause $\neg \tilde{x}_0$ and make it undeletable. Similarly, if x_m is a unary clause in F_2, then we add the unary clause $\tilde{x}_m$ to F', and make it undeletable. Each implicational clause $x_i \to y_j$ in F_2 yields an undeletable chain clause $\alpha \to \beta$ in F', where we define α and β as follows. Let $y_0 \to y_1 \to \cdots \to y_n$ be the ordinary clause in F_2 that contains y_j. Then the corresponding chain clause in F' is

$$y_0 \to y_0' \to \tilde{y}_0 \to y_1 \to y_1' \to y_2 \to y_2' \to \cdots \to y_{n-1} \to y_{n-1}' \to \tilde{y}_n \to y_n \to y_n',$$

and $\alpha = x_i'$, $\beta = y_j$. It is easy to see that there is a deletion set for the modified VDCS problem instance F_2 of size k if and only if there is a deletion set of size k for the $(2\ell+2)$-CDCS instance F', and we can easily get a deletion set F for the VDCS instance from a deletion set for the CDCS instance F'. □

Lemma 5.3. *There is a parameterized reduction from ℓ-CDCS to ℓ-VDCS.*

Proof. Let F be the ℓ-CDCS instance. We produce the desired instance F' of VDCS as follows. For each variable x, let $C_1, \ldots, C_q$ be all the clauses (both unary and chain) that contain x. We make q copies $x_1, \ldots, x_q$ of x, and for each $i \in [q]$, if C_i is a chain clause in F, then it becomes an ordinary clause in F', and if C_i as a unary clause, then C_i remains a unary clause in F', and in addition, an ordinary clause x_i is added to F'. To ensure that the copies of x behave similarly, for each $i, j \in [q], i \neq j$, we add an implicational clause $x_i \to x_j$ to F'.

Removing a chain clause in F corresponds to removing the variables of the corresponding ordinary clause in F'. Removing a unary clause in F also corresponds to removing the variable in the chain clause associated with that unary clause. The converse is equally easy. We can easily produce a deletion set for F from a deletion set for F'.

Note that if $\ell = 1$ in ℓ-CDCS, then we would have a reduction from 1-CDCS to 2-VDCS. We can avoid this blow-up by solving the 1-CDCS instance directly (which is easy), and then producing a 1-VDCS formula requiring the same number of deletions. □

5.2 Reductions

Bipartite graphs whose complement is a circular arc graph admit a simple representation (see [12, 32]).

Definition 5.4. *The class of bipartite graphs whose complement is a circular arc graph corresponds to the class of graphs that can be represented as follows. Let C be a circle, and N*

and S be two different points on C. A northern arc is an arc that contains N but not S. A southern arc is an arc that contains S but not N. Each vertex $v \in V(H)$ is represented by a northern or a southern arc A_v. The pair $\{u, v\}$ is an edge of H if and only if the arcs A_v and A_u do not *intersect.*

First we reduce ℓ-VDCS to DL-HOM(H), where H is a bipartite graph whose complement is a circular arc graph. In fact, we reduce it to the special case FS-FC(H) (making the statement somewhat stronger).

Lemma 5.5. *For every ℓ, there is a bipartite graph H_ℓ whose complement is a circular arc graph such that there is a parameterized reduction from ℓ-VDCS to FS-FC(H_ℓ).*

Proof. Let F be any instance of ℓ-VDCS. We construct in parallel a graph H_ℓ and an instance G_F of DL-HOM(H) such that F is satisfiable after removing the variables of k clauses if and only if G_F maps to H_ℓ after removing k vertices. We will see that the construction of H_ℓ is independent of F and depends only on ℓ.

To define H_ℓ, first fix a circle with two different points N and S. For each ordinary clause C of F, we introduce a vertex $\alpha(C)$ in (the top bipartite class of) G_F. There are at most $\ell+1$ satisfying assignments of the clause C, and therefore we introduce $\ell+1$ arcs $a_0, \dots, a_\ell$ in H_ℓ to encode all these possibilities. To define these arcs, we place $\ell+1$ points $p_0, \dots, p_{\ell-1}, p_\ell$ on the semicircle from S to N in the clockwise direction such that $p_0 \neq S$ and $p_\ell \neq N$. Similarly, we place $\ell+1$ points $q_0, q_1, \dots, q_\ell$ on the semicircle from N to S in the clockwise direction such that $q_0 \neq N$ and $q_\ell \neq S$. The arc a_i goes from p_i to q_i crossing N (but not S), $i \in \{0, \dots, \ell\}$. See Figure 3. We refer to these arcs as *value* arcs.

From any ordinary clause to any other ordinary clause, there are at most ℓ^2 possible implicational clauses. Therefore for all possible pairs (i, j), $0 \leq i, j \leq \ell-1$ we introduce a set of 6 arcs in H_ℓ: $u^1_{i,j}, u^2_{i,j}, v^1_{i,j}, v^2_{i,j}, w^1_{i,j}, w^2_{i,j}$. The role of these sets of arcs is to simulate the implicational clauses as follows. For each implicational clause D in F, let C and C' be the two (distinct) ordinary clauses associated with it. For each such C, C' and D, the graph G_F contains a path $P = \alpha(C) - U(D) - V(D) - W(D) - \alpha(C')$. The clauses C and C' will determine the lists of $\alpha(C)$ and $\alpha(C')$. The clause D will determine i and j, and i and j determine the lists of $U(D), V(D)$ and $W(D)$: $L(U(D)) = \{u^1_{i,j}, u^2_{i,j}\}$, $L(V(D)) = \{v^1_{i,j}, v^2_{i,j}\}$ and $L(W(D)) = \{w^1_{i,j}, w^2_{i,j}\}$.

Suppose that $C = x_0 \to \dots \to x_t$, $C' = y_0 \to \dots \to y_{t'}$, and $D = x_r \to y_{r'}$. We set the list of $\alpha(C)$ to be $\{a_0, \dots, a_{t+1}\}$, and the list of $\alpha(C')$ to $\{a_0, \dots, a_{t'+1}\}$. Finally, we define the lists of $U(D), V(D)$ and $W(D)$. We set the list of $U(D)$ to be $\{u^1_{r,r'}, u^2_{r,r'}\}$, where $u^1_{r,r'}$ is an arc in H_ℓ that starts at S, and goes clockwise to include p_r but not p_{r+1}. The arc $u^2_{r,r'}$ starts at S and goes anticlockwise to a point e that is (strictly) between S and q_ℓ. Arc $v^2_{r,r'}$ starts at N and goes clockwise until it includes e (but not S). Arc $v^1_{r,r'}$ starts at N, it goes anticlockwise to a point f that is (strictly) between p_0 and S. The arc $w^1_{r,r'}$ starts at S and goes clockwise until it includes f but not p_0. Finally, $w^2_{r,r'}$ starts at S and goes anticlockwise until it includes $q_{r'+1}$ but not $q_{r'}$.

Assume now that $\alpha(C)$ is mapped to a value arc a_i such that $i \leq r$. Then a_i intersects $u^1_{r,r'}$, so $U(D)$ must be mapped to $u^2_{r,r'}$. The arc $u^2_{r,r'}$ intersects $v^2_{r,r'}$, so $V(D)$ must be mapped to $v^1_{r,r'}$ which in turn intersects $w^1_{r,r'}$, so $W(D)$ must be mapped to $w^2_{r,r'}$. The arc $w^2_{r,r'}$ intersects any value arc $a_{i'}$ such that $i' > r'$, so $\alpha(C')$ must be mapped to an arc $a_{i'}$ such that $i' \leq r'$.

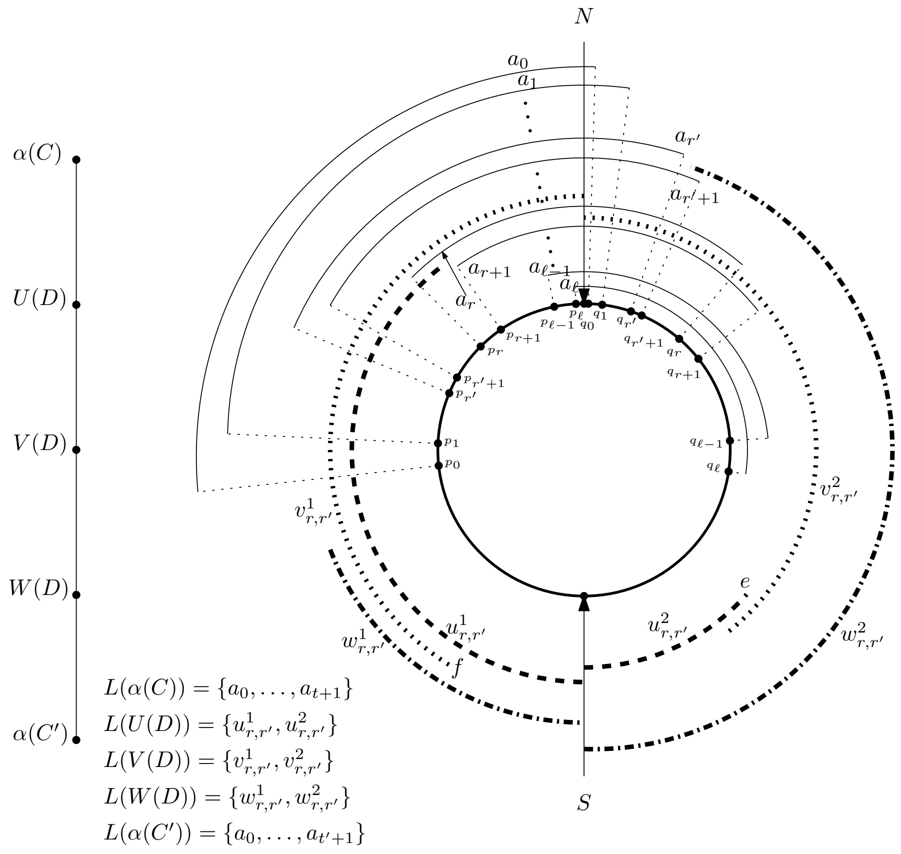

Fig. 3. Construction of the graph H_ℓ and the gadgets in the proof of Lemma 5.5.

On the other hand, if $\alpha(C)$ is mapped to a value arc a_i such that $i > r$, then the above "chain reaction" is not triggered, so $\alpha(C')$ can be mapped to any vertex in its list. More precisely, $U(D)$ can be mapped to $u^1_{r,r'}$, $V(D)$ can be mapped to $v^2_{r,r'}$, and $W(D)$ can be mapped to $w^1_{r,r'}$, which does not intersect any of the value arcs, so $\alpha(C')$ can be mapped to anything in its list.

The above analysis suggests the following correspondence between variable assignments of F and homomorphisms from G_F to H_ℓ. Mapping $\alpha(C)$ to a_i precisely corresponds to the assignment $x_0 = \cdots = x_{i-1} = 0$, $x_i = \cdots = x_q = 1$ of the variables of C. It is clear that using this correspondence, given a satisfying assignment we can construct a homomorphism and vice versa. The unary clauses are encoded using the lists of the variables corresponding to ordinary clauses. For example, if x_i is a unary clause and x_i is among the variables of the ordinary clause C, then in any valid variable assignment we must have that $x_i = 1, x_{i+1} = 1, \ldots, x_q = 1$. In our interpretation, this corresponds to restricting the possible images of $\alpha(C)$ to $a_0, a_1, \ldots, a_i$, so we simply remove the rest of the arcs form $L(\alpha(C))$. Similarly, if $\neg x_i$ is a unary clause, then we must remove a_j from $L(\alpha(C))$ for $j \leq i$.

We give a simple example. Assume that $C = x_0 \to x_1 \to x_2 \to x_3 \to x_4$ and $C' = y_0 \to y_1 \to y_2 \to y_3$ are ordinary clauses and $D = x_2 \to y_2$ is an implicational clause. Then given a satisfying variable assignment such that at least one of x_0, x_1, and x_2 is true, e.g., $x_0 = 0$ and $x_1 = x_2 = 1$, we assign $\alpha(C)$ to a_1. Because we were given a satisfying assignment, we must have that $y_2 = y_3 = 1$. So for example, if y_1 is the first among y_0, y_1, y_2 that has value 1, then we assign $\alpha(C')$ to a_1. We verify that this mapping can be extended to the other 3 vertices of the path between $\alpha(C)$ and $\alpha(C')$. Arc a_1 intersects $u^1_{2,2}$, so $U(D)$ can be assigned (only) to $u^2_{2,2}$. Then $V(D)$ can be assigned (only) to $v^1_{2,2}$, and $W(D)$ (only) to $w^2_{2,2}$, which intersects a_3. Therefore $\alpha(C')$ can be assigned to any of $\{a_0, a_1, a_2\}$ (but not to a_3 or a_4), corresponding to the three possible ways the variables of C' could be assigned. If y_1 is the first having value 1, then we assign $\alpha(C')$ to a_1.

On the other hand, if x_0, x_1, and x_2 are all false, then we assign $\alpha(C)$ to a_i where x_i is the first variable in C with value 1, or if all the variables have value 0, then $i = 5$. In all these cases, the first variable among y_0, y_1, y_2, y_3 that has value 1 could be any of these variables, or all variables could be assigned 0. If the first variable that has value 1 is y_j, then we assign $\alpha(C')$ to a_j. If all variables are 0, we assign $\alpha(C')$ to a_4. We can check that this mapping can be extended to the variables of the path between $\alpha(C)$ and $\alpha(C')$.

For the converse, $\alpha(C)$ and $\alpha(C')$ are prevented by the path between them to be assigned to value arcs that encode a variable assignment violating the conjunction of C, C' and D. If $\alpha(C)$ is mapped to a_i, where $i \leq 2$ (i.e., all those cases where x_2 has value 1), then there is no homomorphism that maps $\alpha(C')$ to a_3 or a_4.

Finally, we need to ensure that only vertices corresponding to ordinary clauses can be removed, i.e., vertices of the form $\alpha(C)$ and $\alpha(C')$. To achieve this, for every path $\alpha(C) - U(D) - V(D) - W(D) - \alpha(C')$ we make the inner vertices "undeletable" as follows. We replace $U(D)$, $V(D)$, and $W(D)$ with $k+1$ copies, where the copies inherit the lists. We add an edge between $\alpha(C)$ and any copy of $U(D)$, an edge between any copy of $U(D)$ and any copy of $V(D)$, an edge between any copy of $V(D)$ and any copy of $W(D)$, and an edge between any copy of $W(D)$ and $\alpha(C')$. This obviously works.

Now we check the correctness of the reduction. Clearly, if after removing k vertices there is a homomorphism from G_F to H_ℓ, then we can remove the corresponding ordinary clauses from the formula and use the homomorphism to define a satisfying assignment. Notice that the parameters are preserved, i.e., that there is a satisfying assignment of the formula F after removing the variables corresponding to k ordinary clauses if and only if there is a homomorphism from G_F to H_ℓ after removing k vertices. Furthermore, we also obtained a deletion set for F from the deletion set for G_F.

Conversely, if there is a satisfying assignment after removing k ordinary clauses, then we remove the corresponding vertices from G_F and define a homomorphism from the satisfying assignment. To do this, note that if either endpoint of a path $\alpha(C) - U(D) - V(D) - W(D) - \alpha(C')$ is removed, say $\alpha(C')$, then for any assignment of the remaining end vertex (e.g., $\alpha(C)$), we can find images for the vertices $U(D)$, $V(D)$ and $W(D)$ such that each edge of $\alpha(C) - U(D) - V(D) - W(D)$ is mapped to an edge of H_ℓ. Clearly, this argument also works when we work with the copies of the inner vertices instead of the originals.

Instance G_F is obviously "fixed side". Since H_ℓ has a single component, G_F is also "fixed component". (Observing that $w^1_{r,r'}$ and $u^2_{r,r'}$ are connected to all the value arcs easily gives that H_ℓ is connected.) □

For the converse direction, the following lemma reduces the special case FS(H) to ℓ-VDCS, where FS(H) is the relaxation of the problem FS-FC(H) where a list could contain vertices from more than one component of H (that is, we only require that the lists are consistent

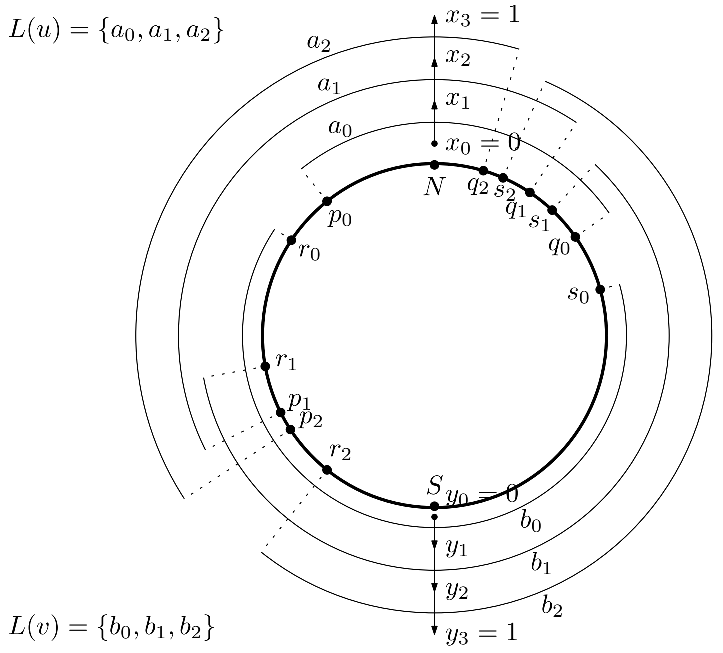

Fig. 4. The graph H in the proof of Lemma 5.6.

fixed side). Together with Theorem 3.1, it gives a reduction from (a general instance of) DL-Hom(H) to ℓ-VDCS.

Lemma 5.6. *Let H be a bipartite graph whose complement is a circular arc graph. Then there is a parameterized reduction from FS(H) to ℓ-VDCS, where $\ell = |V(H)| + 1$.*

Proof. Let G be an instance of the fixed-side problem with bipartition classes T and B, and assume we are given a representation of H as in Definition 5.4, where the special points on the circle are N and S. Let $u \in T$. Clearly, we can assume that no arc $a \in L(u)$ contains any other arc in $L(u)$, as then we can remove a from $L(u)$. Suppose that $t = |L(u)|$, and that the arcs in $L(u)$ are $a_0, \dots, a_{t-1}$ (recall that these arcs contain N but not S; for clarity, sometimes we write a_i^u instead of a_i). Furthermore, let p_i and q_i be points on the circle such that arc a_i is the segment of the circle that begins at p_i, goes clockwise passing N, and ends at point q_i. By renaming the arcs if necessary, we can assume that when we traverse the circle in the clockwise direction starting at p_{t-1}, we visit the endpoints of the arcs in $L(u)$ in the order $p_{t-1}, p_{t-2}, \dots, p_0, q_{t-1}, q_{t-2}, \dots, q_0$. See Figure 4 for an example. Similarly, let $v \in B$ and $t' = |L(v)|$. Let the arcs in $L(v)$ be $b_0, \dots, b_{t'-1}$, and suppose that r_i and s_i are points on the circle such that the arc obtained by going from r_i to s_i in the anticlockwise direction (thus traversing S) is precisely the arc b_i. Just as before, we assume that traversing the circle in the anticlockwise direction starting at r_0, we visit the endpoints of the arcs in $L(u)$ in the order $r_0, r_1, \dots, r_{t'-1}, s_0, s_1, \dots, s_{t'-1}$. Note, however, that the relative ordering of the p_i's and r_i's can be arbitrary; in particular, it is not always true that they strictly alternate (e.g., in Figure 4, we have the ordering p_0, r_0, r_1, p_1, p_2, r_2).

For a vertex $u \in T$, we introduce $t + 1$ ($t = |L(u)|$) variables $x_0, \dots, x_t$ in our VDCS instance. Arc a_i is associated with the pair (x_i, x_{i+1}). We add the ordinary clause $x_0 \to \cdots \to x_t$, and the unary clauses $\neg x_0$, x_t. Notice that if these clauses are satisfied by a variable assignment, then there is a unique index i such that $x_i = 0$ and $x_{i+1} = 1$. Intuitively, this $0-1$ transition indicates that vertex u of G is assigned to a_i. Any vertex $v \in B$ is handled in a similar fashion, i.e., we introduce an ordinary chain clause $y_0 \to \cdots \to y_{t'}$ and unary clauses $\neg y_0$ and $y_{t'}$, where $t' = |L(v)|$. See Figure 4.

We define implicational clauses to encode that edges of G must be mapped to edges of H. Let $\{u, v\}$ be an edge of G. We interpret $x_i = 1$ as u being assigned to one of $a_0, \ldots, a_{i-1}$, and $x_i = 0$ as u being assigned to one of $a_i, \ldots, a_{t-1}$. Similarly for $v \in B$, $y_j = 1$ is interpreted as assigning v to one of $b_0, \ldots, b_{j-1}$, and $y_j = 0$ as v being assigned to $b_j, \ldots, b_{t'-1}$.

We introduce two sets of implicational clauses, the first one consisting of t clauses, and the second one consisting of t' clauses. The role of the first set is to ensure that if u is mapped to an arc a_i, then v is mapped to an arc b_j that does not intersect a_i on the arc from N to S in the clockwise direction. The role of the second set is to ensure that if v is mapped to an arc b_j, then u is mapped to an arc a_i that does not intersect b_j on the arc from N to S in the anticlockwise direction. We define the first set only, as the second set is defined analogously.

The first set of implicational clauses are $x_i \to y_{j_i}$, one for each $i \in \{1, \ldots, t\}$, where j_i is defined as follows. Let j' be the largest integer such that we can get from q_{i-1} to $s_{j'}$ on the circle in the clockwise direction without crossing S. Then $j_i = j' + 1$. In Figure 4, this corresponds to finding the "outermost" arc containing S that "still" does not intersect a_{i-1}. For example, let $i = 2$ in Figure 4. Then $j_i = 2$, and we obtain the implicational clause $x_2 \to y_2$. The rest of the implicational clauses in the figure are $x_1 \to y_1$ and $x_3 \to y_3$. In the example in the figure, the second set of implicational clauses are $y_1 \to x_1$, $y_2 \to y_1$, and $y_3 \to x_3$.

If we have a homomorphism h from G to H, we can construct a satisfying assignment for the VDCS formula by setting the variables as suggested above. That is, if vertex u is in the bipartite class T of G and u is mapped to an arc $a_i^u \in L(u)$ then we set $x_j = 0$ for all $j \leq i$, and $x_j = 1$ for all $j > i$. We similarly set the values of the variables of any chain clause associated with a vertex in the bipartite class B of G. Notice that this assignment automatically satisfies all the chain clauses.

Consider an arbitrary implicational clause $x_i \to y_j$, where x_i is a variable in a chain clause associated with a vertex $u \in T$, and y_j is a variable associated with a vertex $v \in B$. For the sake of contradiction, assume that $x_i = 1$ and $y_j = 0$. The way we assigned the values of the variables indicates that $h(u) \in \{a_0, \ldots, a_{i-1}\}$, and $v \in \{b_j, \ldots, b_{t'}\}$. But because the clause $x_i \to y_j$ exists, we know that $\{u, v\}$ is an edge of G, so we conclude from the definition of $x_i \to y_j$ that each arc $a_0, \ldots, a_{i-1}$ is intersected by the arc b_j, and also by each of $b_{j+1}, \ldots, b_{t'}$ because of the ordering of the arcs. This contradicts the fact that h is a homomorphism.

Conversely, assume that the VDCS formula is satisfied. Then for each vertex of $u \in T$, we find the (unique) index i of the chain clause associated with u such that $x_i = 0$ and $x_{i+1} = 1$, and we define $h(u)$ to be a_i. We similarly define the images of vertices in B. To see that h is a homomorphism, assume for the sake of contradiction that an edge $\{u, v\}$ of G is assigned to a non-edge of H. Let a and b be the arcs to which u and v are assigned, respectively, and let $x_0, \ldots, x_t$ and $y_0, \ldots, y_{t'}$ be the variables associated with u and v, respectively. Then for some i and j, we have that $x_i = 0, x_{i+1} = 1$ and $y_j = 0, y_{j+1} = 1$; from our definition of how to map the vertices of G to $V(H)$, we have that $a = a_i$ and $b = b_j$. Assume without loss of generality that a_i and b_j intersect on the semi-circle from N to S going in the clockwise direction. Find the smallest j' such that $b_{j'}$ intersects a_i on the semi-circle from N to S going in the clockwise direction. Then by definition, the implicational clause $x_{i+1} \to y_{j'}$ is present in the VDCS formula. Since the formula is satisfied, $y_{j'} = 1$, and because $j' \leq j$, $y_j = 1$, a contradiction.

Regarding the parameters and the deletion sets, if there is a homomorphism from G to H after removing k vertices $v_1, \dots, v_k$, then the ℓ'-VDCS formula obtained by removing the variables of the ordinary clauses associated with $v_1, \dots, v_k$ gives exactly the formula obtained from $G[V(G) \setminus \{v_1, \dots, v_k\}]$ directly. The converse works similarly. □

5.3 Approximability

As a consequence of Lemma 5.6 (reduction from FS(H) to ℓ-VDCS), we can obtain a constant-factor approximation algorithm for FS(H). First, let us observe that ℓ-VDCS admits an approximation algorithm with ratio ℓ invoking an algorithm for minimum cut.

Lemma 5.7. *The ℓ-VDCS problem has a polynomial-time approximation algorithm with ratio ℓ.*

Proof. Given an ℓ-VDCS instance I, we obtain a digraph G_I as follows. Vertices of G_I are the variables of I. There is an edge (u, v) from vertex u to vertex v, if $u \to v$ is an implicational clause, or $u = x_i$ and $v = x_j$, where x_i and x_j belong to the same ordinary clause, and $i < j$. Furthermore, we add two new vertices s and t, and for every unary clause forcing a variable w to be 1, we add the edge (s, w), and for every unary clause forcing a variable z to be 0, we add the edge (w, t). Let C be an st-vertex-cutset of minimum size; such a set can be found in polynomial time. Then it is easy to see that the set of those ordinary clauses that contain a variable corresponding to a vertex in C is a valid deletion set W for I: after removing variables that belong to a clause in W, we can assign 1 to every variable that is reachable from 1, and 0 to every other variable. Our approximation algorithm returns this set of ordinary clauses as a solution.

To show that this algorithm has approximation ratio at most ℓ, suppose that U is an optimal solution for I. Then it is easy to verify that the vertices that correspond to the variables of the ordinary clauses in U form an st-vertex-cutset. The size of this cutset is at most $\ell \cdot |U|$. Therefore, the minimum st-vertex-cutset size and size of the optimum solution ℓ-VDCS differ at most by a factor of ℓ, hence the approximation ratio follows. □

Corollary 5.8. *Let H be a bipartite graph whose complement is a circular arc graph. Then there is a polynomial-time approximation algorithm for FS(H) with ratio $|V(H)| + 1$.*

Proof. It follows from Lemmas 5.6 and 5.7 □

Now we have everything at our hands to prove the main approximation result, Theorem 1.3:

Theorem 1.3. *If H is a fixed bipartite graph whose complement is a circular arc graph, then* DL-Hom(H) *is FPA with ratio $|H| + 1$, and the running time of the FPA-algorithm is $f(k, H) \cdot n^{O(1)}$.*

Proof. It follows from the initial iterative compression argument, Theorem 3.1, and Corollary 5.8. □

6 Concluding Remarks

The list homomorphism problem is a widely investigated problem in classical complexity theory. In this work, we initiated the study of this problem from the perspective of parameterized complexity: we have shown that the DL-Hom(H) is FPT for any skew-decomposable graph H parameterized by the solution size and $|H|$, an algorithmic meta-result unifying the fixed-parameter tractability of some well-known problems. To achieve this, we welded together a number of classical and recent techniques from the FPT toolbox in a novel way. Our research suggests many open problems, four of which are:

1. If H is a fixed bipartite graph whose complement is a circular arc graph, is DL-Hom(H) FPT parameterized by solution size? (Conjecture 1.1.)
2. If H is a fixed digraph such that L-Hom(H) does not contain *a circular* N, is DL-Hom(H) FPT parameterized by solution size? The digraphs which do not contain a circular N are precisely those for which L-Hom(H) is in logspace, assuming NL is different from L (cf. [8]).
3. If H is a matching consisting of n edges, is DL-Hom(H) FPT, where the parameter is only the size of the deletion set?
4. Consider DL-Hom(H) for target graphs H in which both vertices with and without loops are allowed. It is known that for such target graphs L-Hom(H) is in P if and only if H is a *bi-arc* graph assuming P is different from NP (cf. [12]), or equivalently, if and only if H has a *majority polymorphism.* If H is a fixed bi-arc graph, is there an FPT reduction from DL-Hom(H) to ℓ-CDCS, where ℓ depends only on $|H|$?

Note that for the first problem, we already do not know if DL-Hom(H) is FPT when H is a path on 7 vertices. (If H is a path on 6 vertices, once we ensured that the instance has fixed-side lists, there is a simple reduction to Almost 2-SAT.) Observe that the third problem is a generalization of the Vertex Multiway Cut problem parameterized only by the cutset. For the fourth problem, we note that the FPT reduction from DL-Hom(H) to CDCS for graphs without loops relies on the fixed-side nature of the lists involved. Since the presence of loops in H makes the concept of a fixed-side list meaningless, it is not clear how to achieve such a reduction.

Acknowledgment We thank the anonymous referees for their comments that helped improve the presentation, and for pointing out a problem that required some technical work to fix.

References

1. Y. Chen, M. Grohe, and M. Grüber. On parameterized approximability. In *Parameterized and Exact Computation, Second International Workshop, IWPEC 2006, Zürich, Switzerland, September 13-15, 2006, Proceedings*, pages 109–120, 2006.
2. R. H. Chitnis, M. Cygan, M. Hajiaghayi, and D. Marx. Directed Subset Feedback Vertex Set Is Fixed-Parameter Tractable. In *Automata, Languages, and Programming - 39th International Colloquium, ICALP 2012, Warwick, UK, July 9-13, 2012, Part I*, pages 230–241, 2012.
3. R. H. Chitnis, L. Egri, and D. Marx. List H-Coloring a Graph by Removing Few Vertices. In *Algorithms - ESA 2013 - 21st Annual European Symposium, Sophia Antipolis, France, September 2-4, 2013. Proceedings*, pages 313–324, 2013.
4. R. H. Chitnis, M. Hajiaghayi, and D. Marx. Fixed-Parameter Tractability of Directed Multiway Cut Parameterized by the Size of the Cutset. *SIAM Journal on Computing*, 42(4):1674–1696, 2013.

5. B. Courcelle. Graph Rewriting: An Algebraic and Logic Approach. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science: Volume B: Formal Models and Semantics*, pages 193–242. Elsevier, Amsterdam, 1990.
6. M. Cygan, M. Pilipczuk, M. Pilipczuk, and J. O. Wojtaszczyk. On Multiway Cut Parameterized Above Lower Bounds. *ACM Transactions on Computation Theory*, 5(1):3, 2013.
7. R. G. Downey and M. R. Fellows. *Fundamentals of Parameterized Complexity.* Texts in Computer Science. Springer, 2013.
8. L. Egri, P. Hell, B. Larose, and A. Rafiey. Space Complexity of List H-colouring: A Dichotomy. In *Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 349–365, 2014.
9. L. Egri, A. A. Krokhin, B. Larose, and P. Tesson. The complexity of the list homomorphism problem for graphs. *Theory of Computing Systems*, 51(2):143–178, 2012.
10. T. Feder and P. Hell. List homomorphisms to reflexive graphs. *Journal of Combinatorial Theory, Series B*, 72(2):236–250, 1998.
11. T. Feder, P. Hell, and J. Huang. List homomorphisms and circular arc graphs. *Combinatorica*, 19(4):487–505, 1999.
12. 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.
13. T. Feder, P. Hell, and J. Huang. List homomorphisms of graphs with bounded degrees. *Discrete Mathematics*, 307:386–392, 2007.
14. T. Feder and M. Y. Vardi. The computational structure of monotone monadic SNP and constraint satisfaction: A study through datalog and group theory. *SIAM Journal on Computing*, 28(1):57–104, 1998.
15. J. Flum, M. Frick, and M. Grohe. Query evaluation via tree-decompositions. *J. ACM*, 49(6):716–752, 2002.
16. J. Flum and M. Grohe. *Parameterized Complexity Theory.* Springer-Verlag, 2006.
17. G. Gutin, A. Rafiey, and A. Yeo. Minimum cost and list homomorphisms to semicomplete digraphs. *Discrete Applied Mathematics*, 154:890–897, 2006.
18. P. Hell and J. Nešetřil. *Graphs and Homomorphisms.* Oxford University Press, 2004.
19. P. Hell and J. Nešetřil. On the Complexity of H-coloring. *Journal of Combinatorial Theory, Series B*, 48:92–110, 1990.
20. P. Hell and A. Rafiey. The Dichotomy of List Homomorphisms for Digraphs. In *Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2011, San Francisco, California, USA, January 23-25, 2011*, pages 1703–1713, 2011.
21. S. Kratsch, M. Pilipczuk, M. Pilipczuk, and M. Wahlström. Fixed-Parameter Tractability of Multicut in Directed Acyclic Graphs. In *Automata, Languages, and Programming - 39th International Colloquium, ICALP 2012, Warwick, UK, July 9-13, 2012, Part I*, pages 581–593, 2012.
22. D. Lokshtanov and D. Marx. Clustering with Local Restrictions. *Information and Computation*, 222:278–292, 2013.
23. D. Lokshtanov, N. S. Narayanaswamy, V. Raman, M. S. Ramanujan, and S. Saurabh. Faster parameterized algorithms using linear programming. *ACM Transactions on Algorithms*, 11(2):15, 2014.
24. D. Lokshtanov and M. S. Ramanujan. Parameterized Tractability of Multiway Cut with Parity Constraints. In *Automata, Languages, and Programming - 39th International Colloquium, ICALP 2012, Warwick, UK, July 9-13, 2012, Part I*, pages 750–761, 2012.
25. D. Marx. Parameterized Graph Separation Problems. *Theoretical Computer Science*, 351(3):394–406, 2006.
26. D. Marx. Parameterized complexity and approximation algorithms. *Comput. J.*, 51(1):60–78, 2008.
27. 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.
28. D. Marx and I. Razgon. Fixed-Parameter Tractability of Multicut Parameterized by the Size of the Cutset. *SIAM Journal on Computing*, 43(2):355–388, 2014.
29. R. Niedermeier. *Invitation to Fixed-Parameter Algorithms.* Oxford University Press, 2006.
30. I. Razgon and B. O'Sullivan. Almost 2-SAT is Fixed-Parameter Tractable. *Journal of Computer and System Sciences*, 75(8):435–450, 2009.
31. B. A. Reed, K. Smith, and A. Vetta. Finding Odd Cycle Transversals. *Operations Research Letters*, 32(4):299–301, 2004.
32. J. Spinrad. Circular-arc Graphs with Clique Cover Number two. *Journal of Combinatorial Theory, Series B*, 44(3):300–306, 1988.