List homomorphism problems for signed graphs

Jan Bok
Computer Science Institute, Faculty of Mathematics and Physics, Charles University, Czech Republic
bok@iuuk.mff.cuni.cz

Richard Brewster
Department of Mathematics and Statistics, Thompson Rivers University, Canada
rbrewster@tru.ca

Tomás Feder
268 Waverley St., Palo Alto, USA
tomas@theory.stanford.edu

Pavol Hell
School of Computing Science, Simon Fraser University, Canada
pavol@cs.sfu.ca

Nikola Jedličková
Department of Applied Mathematics, Faculty of Mathematics and Physics, Charles University, Czech Republic
jedlickova@kam.mff.cuni.cz

Abstract
We consider homomorphisms of signed graphs from a computational perspective. In particular, we study the list homomorphism problem seeking a homomorphism of an input signed graph $(G, \sigma)$, equipped with lists $L(v) \subseteq V(H)$, $v \in V(G)$, of allowed images, to a fixed target signed graph $(H, \pi)$. The complexity of the similar homomorphism problem without lists (corresponding to all lists being $L(v) = V(H)$) has been previously classified by Brewster and Siggers, but the list version remains open and appears difficult. We illustrate this difficulty by classifying the complexity of the problem when $H$ is a tree (with possible loops). The tools we develop will be useful for classifications of other classes of signed graphs, and we illustrate this by classifying the complexity of irreflexive signed graphs in which the unicoloured edges form some simple structures, namely paths or cycles. The structure of the signed graphs in the polynomial cases is interesting, suggesting they may constitute a nice class of signed graphs analogous to the so-called bi-arc graphs (which characterized the polynomial cases of list homomorphisms to unsigned graphs).

2012 ACM Subject Classification Mathematics of computing → Graph theory

Keywords and phrases complexity, dichotomy, graph homomorphism, signed graph

Funding The first author received funding from the European Union’s Horizon 2020 project H2020-MSCA-RISE-2018: Research and Innovation Staff Exchange and from the Charles University Grant Agency project 1580119. The second author was supported by his NSERC Canada Discovery Grant. The fourth and fifth author were also partially supported by the fourth author’s NSERC Canada Discovery Grant. The fifth author was also supported by the Charles University Grant Agency project 1198419 and by the Czech Science Foundation (GA-ČR) project 19-17314J

1 Motivation

We investigate a problem at the confluence of two popular topics – graph homomorphisms and signed graphs. Their interplay was first considered in an unpublished manuscript of Guenin [15], and has since become an established field of study [23].

We now introduce the two topics separately. In the study of computational aspects of graph homomorphisms, the central problem is one of existence – does an input graph...
2 List homomorphism problems for signed graphs

$G$ admit a homomorphism to a fixed target graph $H$? (The graphs considered here are undirected graphs with possible loops but no parallel edges.) This is known as the graph homomorphism problem. It was shown in [18] that this problem is polynomial-time solvable when $H$ has a loop or is bipartite, and is NP-complete otherwise. This is known as the dichotomy of graph homomorphisms (see [19]). The core of a graph $H$ is a subgraph of $H$ with the smallest number of vertices to which $H$ admits a homomorphism; note that such a subgraph is unique up to isomorphism. A graph with a loop has a vertex with a loop as its core, and a (non-empty) bipartite graph has an edge as its core. Thus an equivalent way of stating the graph dichotomy result is that the problem is polynomial-time solvable when the core of $H$ has at most one edge, and is NP-complete otherwise.

Now suppose the input graph $G$ is equipped with lists, $L(v) \subseteq V(H), v \in V(G)$, and we ask if there is a homomorphism $f$ of $G$ to $H$ such that each $f(v) \in L(v)$. This is known as the graph list homomorphism problem. This problem also has a dichotomy of possible complexities [11] – it is polynomial-time solvable when $H$ is a so-called bi-arc graph and is NP-complete otherwise. Bi-arc graphs have turned out to be an interesting class of graphs; for instance, when $H$ is a reflexive graph (each vertex has a loop), $H$ is a bi-arc graph if and only if it is an interval graph [10].

These kinds of complexity questions found their most general formulation in the context of constraint satisfaction problems. The Feder-Vardi dichotomy conjecture [13] claimed that every constraint satisfaction problem with a fixed template $H$ is polynomial-time solvable or NP-complete. After a quarter century of concerted effort by researchers in theoretical computer science, universal algebra, logic, and graph theory, the conjecture was proved in 2017, independently by Bulatov [8] and Zhuk [30]. This exciting development focused research attention on additional homomorphism type dichotomies, including ones for signed graphs [5, 7, 14].

The study of signed graphs goes back to [16, 17], and has been most notably investigated in [25, 26, 27, 28, 29], from the point of view of colourings, matroids, or embeddings. Following Guenin, homomorphisms of signed graphs have been pioneered in [6] and [22]. The computational aspects of existence of homomorphisms in signed graphs – given a fixed signed graph $(H, \pi)$, does an input signed graph $(G, \sigma)$ admit a homomorphism to $(H, \pi)$ – were studied in [5, 14], and eventually a complete dichotomy classification was obtained in [7]. It is surprisingly similar to the second way we stated the graph dichotomy result above, see Theorem 5 and the discussion following it.

Although typically homomorphism problems tend to be easier to classify with lists than without lists (lists allow for recursion to subgraphs), the complexity of the list homomorphism problem for signed graphs appears difficult to classify [3, 7]. If the analogy to (unsigned) graphs holds again, then the tractable cases of the problem should identify an interesting class of signed graphs, generalizing bi-arc graphs. In this paper, we begin the exploration of this concept. We find that there is interesting structure to the tractable cases.

2 Terminology and notation

A signed graph is a graph $G$, with possible loops and multiple edges (at most two loops per vertex and at most two edges between a pair of vertices), together with a mapping $\sigma : E(G) \rightarrow \{+, -\}$, assigning a sign (+ or −) to each edge and each loop of $G$, so that different loops at a vertex have different signs, and similarly for different edges between the same two vertices. For convenience, we shall usually consider an edge to mean an edge or a loop, and to emphasize otherwise we shall call it a non-loop edge. (Thus we say that
each edge of a signed graph has a sign.) We denote a signed graph by \((G, \sigma)\), and call \(G\) its **underlying graph** and \(\sigma\) its **signature**. When the signature name is not needed, we denote the signed graph \((G, \sigma)\) by \(\hat{G}\) to emphasize that it has a signature even though we do not give it a name. We will usually view signs of edges as colours, and call positive edges **blue**, and negative edges **red**. It will be convenient to call a red-blue pair of edges (or loops) with the same endpoint(s) a **bicoloured edge (or loop)**; however, it is important to keep in mind that formally they are two distinct edges (or loops). By contrast, we call edges (and loops) that are not part of such a pair **unicoloured**; moreover, when we refer to an edge as blue or red we shall always mean the edge is unicoloured blue or red. We also call an edge **at least blue** if it is either blue or bicoloured, and similarly for **at least red** edges. Treating a pair of red-blue edges as one bicoloured edge is advantageous in many descriptions, but introduces an ambiguity when discussing walks, since a walk in a signed graph could be seen as a sequences of incident vertices and edges, and so selecting just one edge from a red-blue pair, or it could be interpreted as a sequence of consecutively adjacent vertices, and hence contain some bicoloured edges. This creates particular problem for cycles, since in the former view, a bicoloured edge would be seen as a cycle of length two, with one red edge and one blue edge. In the literature, the former approach is more common, but here we take the latter approach. Of course, the two views coincide if only walks of unicoloured edges are considered. The sign of a closed walk consisting of unicoloured edges \(\hat{G}\) is the product of the signs of its edges. Thus a closed walk of unicoloured edges is **negative** if it has an odd number of negative (red) edges, and **positive** if it has an even number of negative (red) edges. In the case of unicoloured cycles, we also call a positive cycle **balanced** and a negative cycle **unbalanced**. Note that a vertex with a red loop is a cycle with one negative edge, and hence is unbalanced. A **uni-balanced signed graph** is a signed graph without unbalanced cycles, i.e., a signed graph in which all unicoloured cycles (if any) have an even number of red edges. A **anti-uni-balanced signed graph** is a signed graph in which each unicoloured cycle has an even number of blue edges. Thus we have a symmetry to viewing the signs as colours, in particular \(\hat{G}\) is uni-balanced if and only if \(\hat{G}'\), obtained from \(\hat{G}\) by exchanging the colour of each edge, is anti-uni-balanced. We introduce the qualifier “uni-” because the notion of a balanced signed graph is well established in the literature: it means a signed graph without any unbalanced cycles in the classical view, including the two-cycles formed by red-blue pairs of edges. Thus a balanced signed graph is a uni-balanced signed graph without bicoloured edges and loops.

We now define the **switching** operation. This operation can be applied to any vertex of a signed graph and it negates the signs of all its incident non-loop edges. (The signs of loops are unchanged by switching.) We say that two signatures \(\sigma_1, \sigma_2\) of a graph \(\hat{G}\) are **switching equivalent** if we can obtain \((G, \sigma_2)\) from \((G, \sigma_1)\) by a sequence of switchings. In that case we also say that the two signed graphs \((G, \sigma_1)\) and \((G, \sigma_2)\) are switching equivalent. (We note a sequence of switchings may also be realized by negating all the edges of a single edge cut.) In a very formal way, a signed graph is an equivalence class under the switching equivalence, and we sometimes use the notation \(\tilde{G}\) to mean the entire class.

It was proved by Zaslavsky [26] that two signatures of \(G\) are switching equivalent if and only if they define exactly the same set of negative (or positive) cycles. It is easy to conclude that a uni-balanced signed graph is switching equivalent to a signed graph with all edges and loops at least blue, and an anti-uni-balanced signed graph is switching equivalent to a signed graph with all edges and loops at least red.

We now consider homomorphisms of signed graphs. Since signed graphs \(\hat{G}, \hat{H}\) can be viewed as equivalence classes, a homomorphism of signed graphs \(\hat{G}\) to \(\hat{H}\) should be a
List homomorphism problems for signed graphs

A homomorphism of one representative \((G, \sigma)\) of \(\hat{G}\) to one representative \((H, \pi)\) of \(\hat{H}\). It is easy to see that this definition can be simplified by prescribing any fixed representative \((H, \pi)\) of \(\hat{H}\). In other words, we now consider mapping all possible representatives \((G, \sigma')\) of \(\hat{G}\) to one fixed representative \((H, \pi)\) of \(\hat{H}\). At this point, a homomorphism \(f\) of one concrete \((G, \sigma')\) to \((H, \pi)\) is just a homomorphism of the underlying graphs \(G\) to \(H\) preserving the edge colours. Since there are multiple edges, we can either consider \(f\) to be a mapping of vertices to vertices and edges to edges, preserving vertex-edge incidences and edge-colours, as in [23], or simply state that blue edges map to edges that are at least blue, red edges map to edges that are at least red, and moreover bicoloured edges map to bicoloured edges. We follow the second convention. We also say that a signed graph contains a positive edge joining \(x\) and \(y\) if the edge \(xy\) is at least blue, and similarly we say a signed graph contains a negative edge joining \(x\) and \(y\) if the edge \(xy\) is at least red.

Definition 1. We say that a mapping \(f : V(G) \rightarrow V(H)\) is a homomorphism of the signed graph \((G, \sigma)\) to the signed graph \((H, \pi)\), written as \(f : (G, \sigma) \rightarrow (H, \pi)\), if there exists a signed graph \((G, \sigma')\), switching equivalent to \((G, \sigma)\), such that whenever \(uv\) is a positive edge in \((G, \sigma')\), then \((H, \pi)\) contains a positive edge joining \(f(u)\) and \(f(v)\), and whenever \(uv\) is a negative edge in \((G, \sigma')\), then \((H, \pi)\) contains a negative edge joining \(f(u)\) and \(f(v)\).

There is an equivalent alternate definition (see [23]). A homomorphism of the signed graph \((G, \sigma)\) to the signed graph \((H, \pi)\) is a homomorphism \(f\) of the underlying graphs \(G\) to \(H\), such that for any closed walk \(W\) in \((G, \sigma)\) with only unicoloured edges for which the image walk \(f(W)\) has also only unicoloured edges, the sign of \(f(W)\) in \((H, \pi)\) is the same as the sign of \(W\) in \((G, \sigma)\). (In other words, negative closed walks map to negative closed walks.) This definition does not require switching the input graph before mapping it. The equivalence of the two definitions follows from the theorem of Zaslavsky [26] cited above. That result is constructive, and the actual switching required to produce the switching equivalent signed graph \((G, \sigma')\) can be found in polynomial time [23].

We deduce the following fact.

Lemma 2. Suppose \((G, \sigma)\) and \((H, \pi)\) are signed graphs, and \(f\) is a mapping of the vertices of \(G\) to the vertices of \(H\). Then \(f\) is a homomorphism of the signed graph \((G, \sigma)\) to the signed graph \((H, \pi)\) if and only if \(f\) is a homomorphism of the underlying graph \(G\) to the underlying graph \(H\), which moreover maps bicoloured edges of \((G, \sigma)\) to bicoloured edges of \((H, \pi)\), and for any closed walk \(W\) in \((G, \sigma)\) with only unicoloured edges for which the image walk \(f(W)\) has also only unicoloured edges, the signs of \(W\) and \(f(W)\) are the same.

Note that each negative closed walk contains a negative cycle, and in particular an irreflexive tree \((H, \pi)\) has no negative closed walks. Thus if \((H, \pi)\) is an irreflexive tree, then the condition simplifies to having no negative cycle of \((G, \sigma)\) mapped to unicoloured edges in \((H, \pi)\) (because the image would be a positive closed walk). For reflexive trees, the condition requires that no negative cycle of \((G, \sigma)\) maps to a positive closed walk in \((H, \pi)\), and no positive cycle of \((G, \sigma)\) maps to a negative closed walk.

For our purposes, the simpler Definition 1 is sufficient. Note that whether an edge is unicoloured or bicoloured is independent of switching, and that a homomorphism can map a unicoloured edge or loop in \(\hat{G}\) to a bicoloured edge or loop in \(\hat{H}\) but not conversely.

Let \(\hat{H}\) be a fixed signed graph. The homomorphism problem \(\text{S-Hom}(\hat{H})\) takes as input a signed graph \(\hat{G}\) and asks whether there exists a homomorphism of \(\hat{G}\) to \(\hat{H}\). The formal definition of the list homomorphism problems for signed graphs is very similar.
Definition 3. Let $\hat{H}$ be a fixed signed graph. The list homomorphism problem $\text{List-S-Hom}(\hat{H})$ takes an input a signed graph $\hat{G}$ with lists $L(v) \subseteq V(H)$ for every $v \in V(G)$, and asks whether there exists a homomorphism $f$ of $\hat{G}$ to $\hat{H}$ such that $f(v) \in L(v)$ for every $v \in V(G)$.

We note that when $\hat{H}$ and $\hat{H}'$ are switching equivalent signed graphs, then any homomorphism of an input signed graph $\hat{G}$ to $\hat{H}$ is also a homomorphism to $\hat{H}'$, and therefore the problems $\text{S-Hom}(\hat{H})$ and $\text{S-Hom}(\hat{H}')$, as well as the problems $\text{List-S-Hom}(\hat{H})$ and $\text{List-S-Hom}(\hat{H}')$, are equivalent.

We call a signed graph $\hat{H}$ connected if the underlying graph $H$ is connected. We call $\hat{H}$ reflexive if each vertex of $H$ has a loop, and irreflexive if no vertex has a loop. We call $\hat{H}$ a signed tree if $H$, with any existing loops removed, is a tree.

3 More background and connections to constraint satisfaction

We now briefly introduce the constraint satisfaction problems, in the format used in [13]. A relational system $G$ consists of a set $V(G)$ of vertices and a family of relations $R_1, R_2, \ldots, R_k$ on $V(G)$. Assume $G$ is a relational system with relations $R_1, R_2, \ldots, R_k$ and $H$ a relational system with relations $S_1, S_2, \ldots, S_k$, where the arity of the corresponding relations $R_i$ and $S_i$ is the same for all $i = 1, 2, \ldots, k$. A homomorphism of $G$ to $H$ is a mapping $f : V(G) \rightarrow V(H)$ that preserves all relations, i.e., satisfies $(v_1, v_2, \ldots) \in R_i \implies (f(v_1), f(v_2), \ldots) \in S_i$, for all $i = 1, 2, \ldots, k$. The constraint satisfaction problem with fixed template $H$ asks whether or not an input relational system $G$, with the same arities of corresponding relations as $H$, admits a homomorphism to $H$.

Note that when $H$ has a single relation $S$, which is binary and symmetric, then we obtain the graph homomorphism problem referred to at the beginning of Section [4]. When $H$ has a single relation $S$, which is an arbitrary binary relation, we obtain the digraph homomorphism problem [1] which is in a certain sense [13] as difficult to classify as the general constraint satisfaction problem. When $H$ has two relations $+, -$ then we obtain a problem that is superficially similar to the homomorphism problem for signed graphs, except that switching is not allowed. This problem is called the edge-coloured graph homomorphism problem [3], and it turns out to be similar to the digraph homomorphism problem in that it is difficult to classify [5]. On the other hand, the homomorphism problem for signed graphs [5, 7, 14], seems easier to classify, and exhibits a dichotomy similar to the graph dichotomy classification, see Theorem [5].

List homomorphism problems are also special cases of constraint satisfaction problems, as lists can be replaced by unary relations. Consider first the case of graphs. Suppose $H$ is a fixed graph, and form the relational system $H^\#$ with vertices $V(H)$ and the following relations: one binary relation $E(H)$ (this is a symmetric relation corresponding to the undirected edges of the graph $H$), and $2^{|V(H)|} - 1$ unary relations $R_X$ on $V(H)$, each consisting of a different non-empty subset $X$ of $V(H)$. The constraint satisfaction problem with template $H^\#$ has inputs $G$ with a symmetric binary relation $E(G)$ (a graph) and unary relations $S_X, X \subseteq V(H)$, and the question is whether or not a homomorphism exists. If a vertex $v \in V(G)$ is in the relation $R_X$ corresponding to $S_X$, then any mapping preserving the relations must map $v$ to a vertex in $X$; thus imposing the relation $R_X$ on $v \in V(G)$ amounts to setting $L(v) = X$. Therefore the list homomorphism problem for the graph $H$ is formulated as the constraint satisfaction problem for the template $H^\#$.

Such a translation is also possible for homomorphism of signed graphs. Brewster and Graves introduced a useful construction. The switching graph $(H^+, \pi^+)$ has two vertices
6 List homomorphism problems for signed graphs

\(v_1, v_2\) for each vertex \(v\) of \((H, \pi)\), and each edge \(vw\) of \((H, \pi)\) gives rise to edges \(v_1w_1, v_2w_2\) of colour \(\pi(vw)\) and edges \(v_1w_2, v_2w_1\) of the opposite colour. (This definition applies also for loops, i.e., when \(v = w\).) Then each homomorphism of the signed graph \((G, \sigma)\) to the signed graph \((H, \pi)\) corresponds to a homomorphism of the edge-coloured graph \((G, \sigma)\) to the edge-coloured graph \((H^+, \pi^+)\) and conversely. For list homomorphisms of signed graphs, we can use the same transformation, modifying the lists of the input signed graph. If \((G, \sigma)\) has lists \(L(v), v \in V(G)\), then the new lists \(L^+(v), v \in V(G)\), are defined as follows: for any \(x \in L(v)\) with \(v \in V(G)\), we place both \(x_1\) and \(x_2\) in \(L^+(v)\). It is easy to see that the signed graph \((G, \sigma)\) has a list homomorphism to the signed graph \((H, \pi)\) with respect to the lists \(L\) if and only if the edge-coloured graph \((G, \sigma)\) has a list homomorphism to the edge-coloured graph \((H^+, \pi^+)\) with respect to the lists \(L^+\). The new lists \(L^+\) are symmetric sets in \(H^+\), meaning that for any \(x \in V(H), v \in V(G)\), we have \(x_1 \in L^+(v)\) if and only if we have \(x_2 \in L^+(v)\). Thus we obtain the list homomorphism problem for the edge-coloured graph \((H^+, \pi^+)\), restricted to input instances \((G, \sigma)\) with lists \(L\) that are symmetric in \(H^+\).

As above, we can transform this list homomorphism problem for the edge-coloured graph \((H^+, \pi^+)\), to a constraint satisfaction problem with the template \(H^+\) obtained by adding unary relations \(R_X = X\), for sets \(X \subseteq V(H^+)\) that are symmetric in \(H^+\).

We conclude that our problems \(\text{List-S-Hom}(\tilde{H})\) fit into the general constraint satisfaction framework, and therefore it follows from \[8, 30\] that dichotomy holds for problems \(\text{List-S-Hom}(\tilde{H})\). We therefore ask which problems \(\text{List-S-Hom}(\tilde{H})\) are polynomial-time solvable and which are NP-complete.

The solution of the Feder-Vardi dichotomy conjecture involved an algebraic classification of the complexity pioneered by Jeavons \[20\]. A key role in this is played by the notion of a polymorphism of a relational structure \(H\). If \(H\) is a digraph, then a polymorphism of \(H\) is a homomorphism \(f\) of some power \(H^t\) to \(H\), i.e., a function \(f\) that assigns to each ordered \(t\)-tuple \((v_1, v_2, \ldots, v_t)\) of vertices of \(H\) a vertex \(f(v_1, v_2, \ldots, v_t)\) such that two coordinatewise adjacent tuples obtain adjacent images. For general templates, all relations must be similarly preserved.

A polymorphism of order \(t = 3\) is a majority if \(f(v, v, w) = f(v, w, v) = f(w, v, v) = v\) for all \(v, w\). A Siggers polymorphism is a polymorphism of order \(t = 4\), if \(f(a, r, e, a) = f(r, a, r, e)\) for all \(a, r, e\). One formulation of the dichotomy theorem proved by Bulatov \[8\] and Zhuk \[30\] states that the constraint satisfaction problem for the template \(H\) is polynomial-time solvable if \(H\) admits a Siggers polymorphism, and is NP-complete otherwise. Majority polymorphisms are less powerful, but it is known \[13\] that if \(H\) admits a majority then the constraint satisfaction problem for the template \(H\) is polynomial-time solvable. Moreover, we have shown in \[11\] that a graph \(H\) is a bi-arc graph if and only if the associated relational system \(H^+\) admits a majority polymorphism. Thus the list homomorphism problem for a graph \(H\) with possible loops is polynomial-time solvable if \(H^+\) admits a majority polymorphism, and is NP-complete otherwise. It was observed in \[21\] that this is not true for signed graphs.

There is a convenient way to think of polymorphisms \(f\) of the relational system \((H, \pi)^+\). A mapping \(f\) is a polymorphism of \((H, \pi)^+\) if and only if it is a polymorphism of the edge-coloured graph \((H^+, \pi^+)\) and if, for any symmetric set \(X \subseteq V(H^+)\), we have \(v_1, v_2, \ldots, v_t \in X\) then also \(f(v_1, v_2, \ldots, v_t) \in X\). We call such polymorphisms of \((H^+, \pi^+)\) semi-conservative.

We can apply the dichotomy result of \[8, 30\] to obtain an algebraic classification.

**Theorem 4.** For any signed graph \((H, \pi)\), the problem \(\text{List-S-Hom}(H, \pi)\) is polynomial-time solvable if \((H^+, \pi^+)\) admits a semi-conservative Siggers polymorphism, and is NP-complete otherwise.

As mentioned above, one can not replace the semi-conservative Siggers polymorphism by a semi-conservative majority polymorphism \[21\]. We focus in this paper on seeking a graph
theoretic classification, at least for some classes of signed graphs.

4 Basic facts

We first mention the dichotomy classification of the problems $S$-$\text{Hom}(\hat{H})$ from [3]. A subgraph $\hat{G}$ of the signed graph $\hat{H}$ is the s-core of $\hat{H}$ if there is homomorphism $f : \hat{H} \rightarrow \hat{G}$, and every homomorphism $\hat{G} \rightarrow \hat{G}$ is a bijection on $V(G)$. The letter $s$ stands for signed. It is again easy to see that the s-core is unique up to isomorphism and switching equivalence.

▶ Theorem 5. [3] The problem $S$-$\text{Hom}(H, \pi)$ is polynomial-time solvable if the s-core of $(H, \pi)$ has at most two edges, and is NP-complete otherwise.

When the signature $\pi$ has all edges positive, the problem $S$-$\text{Hom}(H, \pi)$ is equivalent to the unsigned graph homomorphism problem, and the s-core of $(H, \pi)$ is just the core of $H$. To compare Theorem 5 with the graph dichotomy theorem, as discussed at the beginning Section 1, we observe that the core of a graph cannot have exactly two edges, as a core must be either a single vertex (possibly with a loop), or a single edge, or a graph with at least three edges. Thus Theorem 5 is stronger than the graph dichotomy theorem, which states that the graph homomorphism problem to $H$ is polynomial-time solvable if the core of $H$ has at most one edge and is NP-complete otherwise. (However, we note that the proof of Theorem 5 in [3] uses the graph dichotomy theorem.)

Observe that an instance of the problem $S$-$\text{Hom}(\hat{H})$ can be also viewed as an instance of $\text{List-S-Hom}(\hat{H})$ with all lists $L(v) = V(H)$, therefore if $S$-$\text{Hom}(\hat{H})$ is NP-complete, then so is $\text{List-S-Hom}(\hat{H})$. Moreover, if $\hat{H}'$ is an induced subgraph of $\hat{H}$, then any instance of $\text{List-S-Hom}(\hat{H}')$ can be viewed as an instance of $\text{List-S-Hom}(\hat{H})$ (with the same lists), therefore if the problem $\text{List-S-Hom}(\hat{H}')$ is NP-complete, then so is the problem $\text{List-S-Hom}(\hat{H})$. This yields the NP-completeness of $\text{List-S-Hom}(\hat{H})$ for all signed graphs $\hat{H}$ that contain an induced subgraph $\hat{H}'$ whose s-core has more than two edges. Furthermore, when the signed graph $\hat{H}$ is uni-balanced, then we may assume that all edges are at least blue, and the list homomorphism problem for $H$ can be reduced to $\text{List-S-Hom}(\hat{H})$. In particular, we emphasize that $\text{List-S-Hom}(\hat{H})$ is NP-complete if $\hat{H}$ is a uni-balanced signed graph (or, by a symmetric argument, an anti-uni-balanced signed graph), and the underlying graph $H$ is not a bi-arc graph [11].

Next we focus on the class of signed graphs that have no bicoloured loops and no bicoloured edges. In this case, the following simple dichotomy describes the classification. (This result was previously announced in [3].) It follows from our earlier remarks that these signed graphs are balanced if and only if they are uni-balanced, and similarly they are anti-balanced if and only if they are anti-uni-balanced.

▶ Theorem 6. Suppose $\hat{H}$ is a connected signed graph without bicoloured loops and edges. If the underlying graph $H$ is a bi-arc graph, and $\hat{H}$ is balanced or anti-balanced, then the problem $\text{List-S-Hom}(\hat{H})$ is polynomial-time solvable. Otherwise, the problem is NP-complete.

Proof. The polynomial cases follow from Feder et al. [11], since balanced or anti-balanced signed graphs can be assumed to have all edges blue, or all red. Otherwise, the graph $\hat{H}$ must contain a cycle which cannot be switched to a blue cycle and a cycle which cannot be switched to a red cycle, in which case the s-core of $\hat{H}$ contains at least three edges. (This is true even if the cycles are just loops.)

We have observed that $\text{List-S-Hom}(\hat{H})$ is NP-complete if the s-core of $\hat{H}$ has more than two edges. Thus we will focus on signed graphs $\hat{H}$ whose s-cores have at most two
edges. This is not as simple as it sounds, as there are many complex signed graphs with this property, including, for example, all irreflexive bipartite signed graphs that contain a bicoloured edge, and all signed graphs that contain a bicoloured loop. That these cases are not easy underlines the fact that the assumptions in Theorem 5 cannot be weakened without significant new breakthroughs. Consider, for example, allowing bicoloured edges but not bicoloured loops. In this situation, we may focus on the case when there is a bicoloured edge (else Theorem 6 applies), and so if there is any loop at all, the s-core would have more than two edges. Thus we consider irreflexive signed graphs, and the s-core is still too big if the underlying graph has an odd cycle. So in this case it remains to classify the irreflexive bipartite signed graphs that contain a bicoloured edge. Even this case is complex, as we discuss in the final section.

In this paper we focus on List-S-Hom(\(\hat{H}\)) when the underlying graph of \(\hat{H}\) is a tree with possible loops. (We have treated the special cases of reflexive and irreflexive trees in the conference presentations [24, 25] .) We also study an additional class of irreflexive bipartite signed graphs \(\hat{H}\) in which the unicoloured edges span a Hamiltonian path. We classify the complexity of these graphs; the structure of these graphs turns out to be surprisingly complex.

We now introduce our basic tool for proving NP-completeness.

**Definition 7.** Let \((U, D)\) be two walks in \(\hat{H}\) of equal length, say \(U\), with vertices \(u = u_0, u_1, \ldots, u_k = v\) and \(D\), with vertices \(u = d_0, d_1, \ldots, d_k = v\). We say that \((U, D)\) is a chain, provided \(u_0, u_{k-1}v\) are unicoloured edges and \(ud_1, u_{k-1}v\) are bicoloured edges, and for each \(i, 1 \leq i \leq k - 2\), we have
1. both \(u_iu_{i+1}\) and \(d_id_{i+1}\) are edges of \(\hat{H}\) while \(d_iu_{i+1}\) is not an edge of \(\hat{H}\), or
2. both \(u_iu_{i+1}\) and \(d_id_{i+1}\) are bicoloured edges of \(\hat{H}\) while \(d_iu_{i+1}\) is not a bicoloured edge of \(\hat{H}\).

**Theorem 8.** If a signed graph \(\hat{H}\) contains a chain, then List-S-Hom(\(\hat{H}\)) is NP-complete.

**Proof.** Suppose that \(\hat{H}\) has a chain \((U, D)\) as specified above. We shall reduce from NOT-ALL-EQUAL SAT. (Each clause has three unnegated variables, and we seek a truth assignment in which at least one variable is true and at least one is false, in each clause.) For each clause \(x \lor y \lor z\), we take three vertices \(x, y, z\), each with list \(\{u\}\), and three vertices \(x', y', z'\), each with list \(\{v\}\). For the triple \(x, y, z\), we add three new vertices \(p(x, y), p(y, z), p(z, x)\), each with list \(\{u, d_1\}\), and for the triple \(x', y', z'\), we add three new vertices \(p(x', y'), p(y', z'), p(z', x')\), each with list \(\{u_{k-1}, d_{k-1}\}\). We connect these vertices as follows:
- \(p(x, y)\) adjacent to \(x\) by a red edge and to \(y\) by a blue edge,
- \(p(y, z)\) adjacent to \(y\) by a red edge and to \(z\) by a blue edge,
- \(p(z, x)\) adjacent to \(z\) by a red edge and to \(x\) by a blue edge.

Analogously, the hexagon \(x', p(x', y'), y', p(y', z'), z', p(z', x')\) will also be alternating in blue and red colours, with (say) \(p(x', y')\) adjacent to \(x'\) by a red edge.

Moreover, we join each pair of vertices \(p(x, y)\) and \(p(x', y')\) by a separate path \(P(x, y)\) of length \(k - 1\), say \(p(x, y) = a_1, a_2, \ldots, a_{k-2}, a_{k-1} = p(x', y')\), where \(a_i\) has list \(\{u, d_i\}\) and the edge \(a_ia_{i+1}\) is blue unless both \(u_{i+1}u_i\) and \(d_{i+1}d_i\) are bicoloured, in which case \(a_ia_{i+1}\) is also bicoloured. Paths \(P(z, x)\) and \(P(y, z)\) are defined analogously. See Figure 1 for an illustration.

We observe for future reference that the path \(x, p(x, y) = a_1, a_2, \ldots, a_{k-2}, a_{k-1} = p(x', y'), x'\), when considered by itself, admits a list homomorphism both to \(U\) and to \(D\), but no list homomorphism to any other subgraph of \(U \cup D\). (To see the first part, invoke Zaslavsky’s theorem characterizing switching equivalent signatures, and use the fact that
both $U$ and $D$ contain a bicoloured edge. To see the second part, use the conditions in the definition of chain.)

If $x$ occurs in more clauses, we link the occurrences by a new vertex $p(x)$ with the list $\{u_i\}$ and blue edges to all occurrences of $x$. (This will ensure that all occurrences of $x$ take on the same truth value.)

We denote the resulting graph $(G, \sigma)$. We now claim that this instance of NOT-ALL-EQUAL SAT is satisfiable if and only if $(G, \sigma)$ admits a list homomorphism to $(H, \pi)$.

Let $\tilde{G}(x, y, z)$ denote the subgraph of $(G, \sigma)$ induced by $P(x, y), P(z, x), P(y, z)$ and $x, y, z, x', y', z'$. We claim that

(i) any list homomorphism of $\tilde{G}(x, y, z)$ to $U \cup D$ must switch at either one or two of the vertices $x, y, z$, and that

(ii) there are list homomorphisms of $\tilde{G}(x, y, z)$ to $U \cup D$ that switch at any one or any two of the vertices $x, y, z$.

Once this claim is proved, we can associate with every truth assignment a list homomorphism of $\tilde{G}(x, y, z)$ to $U \cup D$ where a vertex corresponding to a variable is switched if and only if that variable is true, and conversely, setting a variable true if its corresponding vertex was switched in the list homomorphism.

We now prove (i). Since the lists are so restrictive, any list homomorphism is fully described by what happens to the paths $P(x, y), P(z, x), P(y, z)$, and whether or not the vertices corresponding to $x, y, z$ (and $x', y', z'$) are switched. Note that $U$ begins with a unicoloured edge and $D$ ends with a unicoloured edge. It follows that in any list homomorphism of $\tilde{G}(x, y, z)$ to $U \cup D$ after the necessary switching we must have either the edges $xp(x, y)$ and $p(x, y)y$ of the same colour, or the edges $x'p(x', y')$ and $p(x', y')y'$ of the same colour. (In the former case, we map $P(x, y)$ to $U$, in the latter case, we map it to $D$.) If none of the vertices $x, y, z$ were switched, then the edges $x'p(x', y')$ and $p(x', y')y'$ must be of the same colour, and by a similar argument so must the edges $y'p(y', z')$ and $p(y', z')z'$, as well as $z'p(z', x')$ and $p(z', x')x'$. This would mean that the hexagon $x', p(x', y'), y', p(y', z'), z', p(z', x')$ has an even number of red and even number of blue edges, which is impossible. (It started with an odd number of each.) Similarly, if all three vertices $x, y, z$ were switched, we would obtain the same contradiction.

For (ii), it remains to show that one or two of the vertices $x, y, z$ can be switched under a list homomorphism of $\tilde{G}(x, y, z)$ to $U \cup D$. Suppose first that only one was switched; by symmetry assume it was $x$ (so $y$ and $z$ were not switched). Now edges $x, p(x, y)$ and $p(x, y), y$ have the same colour, and $z, p(z, x)$ and $p(z, x), x$ have the same colour. By the above observation, we can map $P(x, y)$ and $P(z, x)$ to $U$, and map $P(y, z)$ to $D$. Note that the switchings necessary for these list homomorphisms affect disjoint sets of vertices (the

![Figure 1 The clause gadget for clause $(x \lor y \lor z)$ in Theorem 8](image-url)
paths \( P(x, y), P(y, z), P(z, x) \), so the observation applies. If two vertices, say \( y \) and \( z \) were switched, the argument is almost the same and we omit it.

5 Irreflexive trees

In this section, \( \hat{H} \) will always be an irreflexive tree. As trees do not have any cycles, \( \hat{H} \) is trivially uni-balanced, and hence we may assume that all edges are at least blue.

Lemma 9. If the underlying graph \( H \) contains the graph \( F_1 \) in Figure 2, then \( \text{List-S-Hom}(\hat{H}) \) is NP-complete.

Proof. If the underlying graph \( H \) contains the graph \( F_1 \) in Figure 2, then \( H \) is not a bi-arc graph by [11], whence \( \text{List-S-Hom}(\hat{H}) \) is NP-complete by the remarks following Theorem 5.

Lemma 10. If \( \hat{H} \) contains one of the signed graphs in family \( F \) from Figure 3 as an induced subgraph, then \( \text{List-S-Hom}(\hat{H}) \) is NP-complete.

Proof. For each of the signed graphs in family \( F \) we can apply Theorem 8. The figure lists a chain for each of these forbidden subgraphs. Thus any signed graph \( \hat{H} \) that contains one of them as an induced subgraph has NP-complete \( \text{List-S-Hom}(\hat{H}) \).

An irreflexive tree \( H \) is a 2-caterpillar if it contains a path \( P = v_1v_2\ldots v_k \), such that each vertex of \( H \) is either on \( P \), or is a child of a vertex on \( P \), or is a grandchild of a vertex on \( P \), i.e., is adjacent to a child of a vertex on \( P \). We also say that \( H \) is a 2-caterpillar with respect to the spine \( P \). (Note that the same tree \( H \) can be a 2-caterpillar with respect to different spines \( P \).) In such a situation, let \( T_1, T_2, \ldots, T_\ell \) be the connected components of \( H \setminus P \). Each \( T_i \) is a star adjacent to a unique vertex \( v_j \) on \( P \). The tree \( T_i \) together with the edge joining it to \( v_j \) is called a rooted subtree of \( H \) (with respect to the spine \( P \)), and is considered to be rooted at \( v_j \). Note that there can be several rooted subtrees with the same root vertex \( v_j \) on the spine, but each rooted subtree at \( v_j \) contains a unique child of \( P \) (and possibly no grandchildren, or possibly several grandchildren).

If \( H \) is a 2-caterpillar with respect to the spine \( P \), and additionally the bicoloured edges of \( \hat{H} \) form a connected subgraph, and there exists an integer \( d \), with \( 1 \leq d \leq k \), such that:

- all edges on the path \( v_1v_2\ldots v_d \) are bicoloured, and all edges on the path \( v_dv_{d+1}\ldots v_k \) are blue,
- the edges of all subtrees rooted at \( v_1, v_2, \ldots, v_{d-1} \) are bicoloured, except possibly edges incident to leaves, and
- the edges of all subtrees rooted at \( v_{d+1}, \ldots, v_k \) are all blue,

then we call \( \hat{H} \) a good 2-caterpillar with respect to \( P = v_1v_2\ldots v_k \).

The vertex \( v_d \) is called the dividing vertex of \( \hat{H} \). Note that the subtrees rooted at \( v_d \) are not limited by any condition except the connectivity of the subgraph formed by the bicoloured edges. A typical example of a good 2-caterpillar is depicted in Figure 4.

![Figure 2](image-url) The subgraph \( F_1 \).
\begin{align*}
\text{a)} & \quad U = 2 - 3 - \ldots - k - (k - 1) \\
& \quad D = 2 - 1 - 2 - \ldots - (k - 1) \\
\text{b)} & \quad U = 3 - 2 - 1 - 2 - \ldots - (k - 2) \\
& \quad D = 3 - \ldots - (k - 1) - k - (k - 1) - (k - 2) \\
\text{c)} & \quad U = 4 - 8 - 9 - 8 - 4 - 5 - 6 - 7 - 6 - 5 - 4 \\
& \quad D = 4 - 3 - 2 - 1 - 2 - 3 - 4 - 8 - 9 - 8 - 4 \\
\text{d)} & \quad U = 4 - 3 - 2 - 1 - 2 - 3 - 4 - 8 - 4 \\
& \quad D = 4 - 8 - 4 - 5 - 6 - 7 - 6 - 5 - 4 \\
\text{e)} & \quad U = 8 - 4 - 5 - 6 - 7 - 6 - 5 - 4 - 8 - 9 - 8 \\
& \quad D = 8 - 9 - 8 - 4 - 3 - 2 - 1 - 2 - 3 - 4 - 8
\end{align*}

**Figure 3** The family $\mathcal{F}$ of signed graphs yielding NP-complete problems, and a chain in each.

> **Lemma 11.** Let $\hat{H}$ be an irreflexive signed tree. Then $\hat{H}$ is a good 2-caterpillar if and only if it does not contain any of the graphs from family $\mathcal{F}$ in Figure 3 as an induced subgraph, and the underlying graph $H$ does not contain the graph $F_1$ in Figure 2.

**Proof.** It is easy to check that none of the depicted signed graphs admits a suitable spine, and hence they are not good 2-caterpillars. So assume $\hat{H}$ does not contain any of the graphs in Figure 3 as an induced subgraph, and the underlying graph $H$ does not contain the graph $F_1$ in Figure 2 as a subgraph. If $H$ is not a 2-caterpillar with respect to any spine, then the underlying graph $H$ contains the tree in Figure 2. The bicoloured edges of $\hat{H}$ induce a connected subgraph, since there is no subgraph from graph class a) in Figure 3. There exists a path $P = v_1 v_2 \ldots v_k$ with a dividing vertex $v_d$ as specified, because there is no subgraph from graph class b) in Figure 3. The absence of classes b) and c) from Figure 3 also ensures that, there is a suitable spine $P = v_1 v_2 \ldots v_k$ with bicoloured edges on $v_1 v_2 \ldots v_d$ and blue edges on $v_d v_{d+1} \ldots v_k$, and with all subtrees of height two rooted at $v_1, \ldots, v_{d-1}$ attached to $P$ with a bicoloured edge. (For this, we note that in the case c), as long as the edges 34 and 45 are bicoloured, the subtree still yields an NP-complete problem even with any of the edges 12, 23, 56, 67 unicoloured.) The absence of graphs d) and e) in Figure 3 ensures that all subtrees rooted at $v_{d+1}, \ldots, v_k$ have all edges blue.

> **Theorem 12.** Let $\hat{H}$ be an irreflexive tree. If $\hat{H}$ is a good 2-caterpillar, then $\text{List-S-Hom}(\hat{H})$ is polynomial-time solvable. Otherwise, $H$ contains a copy of $F_1$, or $\hat{H}$ contains one of the signed graphs in family $\mathcal{F}$ as an induced subgraph, and the problem is NP-complete.

The second claim follows from Lemmas 9, 10 and 11. We prove the first claim in a sequence of lemmas. Suppose that $\hat{H}$ is a good 2-caterpillar with respect to the spine
Let us first observe that any bipartite min ordering, i.e., that each vertex has its neighbours joined to it by bicoloured edges ordered between it and its unicoloured neighbours, both in $P$.

We distinguish four types of rooted subtrees of $\hat{H}$ with respect to the spine $P$.

- Type $T_1$: a bicoloured edge $v_iv_x$.
- Type $T_2$: a bicoloured edge $v_ix$, bicoloured edges $xz_j$ for a set of vertices $z_j$, and blue edges $xt_j$ for another set of vertices $t_j$.
- Type $T_3$: a blue edge $v_ix$ and blue edges $xt_j$ for a set of vertices $t_j$; and
- Type $T_4$: a blue edge $v_ix$.

In the types $T_2$ and $T_3$ we assume that they are not of type $T_1$ or $T_4$, i.e., that at least some $z_j$ or $t_j$ exist; but we allow in $T_2$ either the set of $z_j$ or the set of $t_j$ to be empty.

Recall that we assume that all edges of $H$ are at least blue. Since the underlying graph $H$ is bipartite, we have also distinguished its vertices as black and white; we assume that $v_1$ is white.

A bipartite min ordering of the bipartite graph $H$ is a pair $<_b,<_w$, where $<_b$ is a linear ordering of the black vertices and $<_w$ is a linear ordering of the white vertices, such that for white vertices $x<_w x'$ and black vertices $y<_b y'$, if $xy', x'y$ are both edges in $H$, then $xy$ is also an edge in $H$. It is known \cite{13} that if a bipartite graph $H$ has a bipartite min ordering, then the list homomorphism problem for $H$ can be solved in polynomial time as follows. First apply the arc consistency test, which repeatedly visits edges $xy$ and removes from $L(x)$ any vertex of $H$ not adjacent to some vertex of $L(y)$, and similarly removes from $L(y)$ any vertex of $H$ not adjacent to some vertex of $L(x)$. After arc consistency, if there is an empty list, no list homomorphism exists, and if all lists are non-empty, choosing the minimum element of each list, according to $<_b$ or $<_w$, defines a list homomorphism as required. We call a bipartite min-ordering of the signed irreflexive tree $\hat{H}$ special if for black vertices $x, x'$ and white vertices $y, y'$, if $xy$ is bicoloured and $xy'$ is blue, then $y<_w y'$, and if $xy$ is bicoloured and $x'y$ is blue, then $x<_b x'$. In other words, the bicoloured neighbours of any vertex appear before its unicoloured neighbours, both in $<_b$ and in $<_w$.

> **Lemma 13.** Every good 2-caterpillar $\hat{H}$ admits a special bipartite min ordering.

**Proof.** Let us first observe that any 2-caterpillar admits a bipartite min ordering $<_b,<_w$ in which $v_1<_w v_3<_w v_5, \ldots$ and $v_2<_b v_4<_b v_6, \ldots$, and each subtree rooted at a vertex $v_i$ has the children of $v_i$ ordered between $v_{i-1}$ and $v_{i+3}$ and the grandchildren of $v_i$ ordered between $v_i$ and $v_{i+2}$. We only need to ensure that the order of the grandchildren conforms to the order of the children, i.e., if a child $a$ of $v_i$ is ordered before a child $b$ of $v_i$ then the children of $a$ are all ordered before the children of $b$. Also, all children of $v_i$ are ordered after all grandchildren of $v_{i-1}$. See Figure 5 for an illustration.

It remains to ensure that the bipartite min ordering we choose is in fact a special bipartite min ordering, i.e., that each vertex has its neighbours joined to it by bicoloured edges ordered

![Figure 5](An example of special bipartite min ordering.)
before its neighbours joined to it by unicoloured edges. Therefore the subtrees rooted at each \(v_i\) are treated as follows. We will order first vertices of subtrees of type \(T_1\) one at a time, then vertices of subtrees of type \(T_2\) one at a time, then vertices of subtrees of type \(T_3\) one at a time, and finally vertices of subtrees of type \(T_4\) one at a time. Each subtree of type \(T_1\) consists of only one bicoloured edge, and we order these consecutively between \(v_{i-1}\) and \(v_{i+1}\). Next in order will come the children of \(v_i\) in subtrees of type \(T_2\), still before \(v_{i+1}\), and in each of these subtrees we order first the grandchildren of \(v_i\) incident to a bicoloured edge before those adjacent to a unicoloured edge. We order the subtrees of type \(T_3\) similarly. Finally, for subtrees of type \(T_4\), we order their vertices (each a child of \(v_i\)) right after \(v_{i+1}\).

Lemma 14. If a signed irreflexive graph \(\hat{H}\) admits a special bipartite min ordering, then \(\text{List-S-Hom}(\hat{H})\) is polynomial-time solvable.

Proof. We describe a polynomial-time algorithm. Suppose \(\hat{G}\) is the input signed graph; we may assume \(\hat{G}\) is connected, bipartite, and such that the black vertices have lists with only the black vertices of \(\hat{H}\), and similarly for the white vertices. The first step is to perform the arc consistency test for the existence of a homomorphism of the underlying graphs \(G\) to \(H\), using the special bipartite min ordering \(<_h,<_w>\). We also perform bicoloured arc consistency test, which repeatedly visits bicoloured edges \(xy\) of \(G\) and removes from \(L(x)\) any vertex of \(H\) not adjacent to some vertex of \(L(y)\) by a bicoloured edge, and similarly removes from \(L(y)\) any vertex of \(H\) not adjacent to some vertex of \(L(x)\) by bicoloured edge. If it yields an empty list, there is no list homomorphism of the underlying graphs, and hence no list homomorphism of signed graphs. Otherwise, the minima of all lists define a list homomorphism \(f: G \to H\) of the underlying graphs, by \([13]\). By the bicoloured arc consistency test, the minimum choices imply that the image of a bicoloured edge under \(f\) is also a bicoloured edge. According to Lemma 2 and the remark following it, \(f\) is also a list homomorphism of signed graphs unless a negative cycle \(C\) of unicoloured edges of \(\hat{G}\) maps to a closed walk \(f(C)\) of blue edges in \(\hat{H}\).

Now we make use of the properties of special bipartite min ordering to repair the situation, if possible. Note that the fact that we choose minimum possible values for \(f\) means that we cannot map \(C\) lower in the orders \(<_h,<_w>\). We consider three possible cases.

- **At least one of the edges of \(f(C)\) is in a subtree \(T\) of type \(T_2\) rooted at some \(v_i\), with \(i \leq d\):**
  In this case, all edges of \(f(C)\) must be in \(T\), since the edge of \(T\) incident to \(v_i\) is bicoloured. Assume without loss of generality that \(v_i\) is white, \(x\) is the unique child of \(v_i\) in \(T\), and \(xt_1,\ldots,xt_m\) are the blue edges of \(T\), where \(x\) is black and \(t_1,\ldots,t_m\) are white. Since \(f(C)\) is included in the edges \(xt_1,\ldots,xt_m\) and \(v_i\) precedes in \(<_w\) all vertices \(t_1,\ldots,t_m\), the final lists of the white vertices in \(C\) do not include \(v_i\) (since we assigned the minimum value in each list). Therefore under any homomorphism the image of the connected graph \(C\) either is included in the set of edges \(xt_1,\ldots,xt_m\), or is disjoint from this set of edges. Since we have already explored the first possibility, we can delete the vertices \(t_1,\ldots,t_m\) from the lists of all white vertices of \(C\) and repeat the arc consistency test. This will check whether there is possibly another list homomorphism of graphs \(G \to H\), which is also a homomorphism of signed graphs \(\hat{G} \to \hat{H}\).

- **At least one of the edges of \(f(C)\) is in a subtree of type \(T_4\) rooted at some \(v_i, i \leq d - 1\):**
  In this case, all edges of \(f(C)\) must be in subtrees of type \(T_4\) rooted at the same \(v_i\). Assume again, without loss of generality, that \(v_i\) is white and the subtrees consist of the blue edges \(v_ix_1,v_ix_2,\ldots,v_ix_m\), with each \(x_j\) black. Since \(<_b,<_w>\) is a special bipartite min ordering, all vertices adjacent to \(v_i\) by a bicoloured edge are smaller in \(<_b>\) than \(x_1,\ldots,x_m\). Therefore no such vertex can be in a list of a black vertex in \(C\). This again
means that the image of $C$ is either included in the set of edges $v_i x_1, v_i x_2, \ldots, v_i x_m$, or is disjoint from this set of edges. We can delete all vertices $x_1, \ldots, x_m$ from the lists of all black vertices of $C$ and repeat as above.

The edges of $f(C)$ are included in the set of the edges on the path $v_d v_{d+1} \ldots v_k$ and in the subtrees of types $T_3$ or $T_4$ rooted at $v_d, \ldots, v_k$.

In this case, the vertices of the cycle $C$ have lists containing only vertices incident with blue edges, and there is no homomorphism of signed graphs $\hat{G} \to \hat{H}$.

After we modified the image of one negative cycle $C$ of $\hat{H}$, we proceed to modify another, until we either obtain a homomorphism of signed graph, or find that no such homomorphism exists. The algorithm is polynomial, because arc consistency can be performed in linear time [13], and each modification removes at least one vertex of $H$ from the list of at least one vertex of $G$. Recall that the graph $H$ is fixed, and hence its number of vertices is a constant $k$. If $G$ has $n$ vertices, then this step will be performed at most $kn$ times.

6 Reflexive trees

We now turn to reflexive trees, and hence in this section, $\hat{H}$ will always be a reflexive tree. We may have red, blue, or bicoloured loops, but we may again assume that all non-loop unicoloured edges are of the same colour (blue or red).

Lemma 15. If $\hat{H}$ contains one of the graphs from the family $\mathcal{G}$ in Figure 6 as an induced subgraph, then List-S-Hom($\hat{H}$) is NP-complete.

Proof. The signed graphs in a), b) and c) are themselves $s$-cores with more than two edges, so it follows from Theorem 5 that they yield NP-complete cases.

The cases d) to l) (in fact, c) as well) have chains as is described in Figure 6.

In the final case m) we reduced Not-All-Equal SAT to List-S-Hom($H, \pi$) where $(H, \pi)$ is the signed graph in Figure 6 m). Let $(T, \sigma)$ be the signed graph with the list assignments and signature shown in Figure 7. For each clause $(x, y, z)$ in the instance of Not-All-Equal SAT, we create a copy of $(T, \sigma)$ identifying the leaves $x, y, z$ in $T$ with the variables in the clause.

We claim that $(T, \sigma)$ admits a list homomorphism to $(H, \pi)$ if and only if we switch at exactly one or two elements of $\{x, y, z\}$. Consider a mapping of $(T, \sigma)$ to $(H, \pi)$. It is easy to see that either both $x$ and $m$ are switched or neither is switched. It is also easy to see if $m$ maps to 2, then exactly one of $m$ or $y$ must be switched. Conversely, if $m$ maps to 4, then neither or both of $m$ and $y$ is switched. (In the former case the image of the path is a negative walk, while in the latter case it is positive.) Hence, when $m$ maps to 2, exactly one of $x$ or $y$ is switched, and when $m$ maps to 4, either both or neither $x$ and $y$ is switched. Finally, if $m$ maps to 2, then we may choose to switch or not switch at $z$. On the other hand, if $m$ maps to 4, then we must switch at $z$ if and only if we do not switch at $m$. Viewing switching at a variable as setting the variable to true and not switching as setting the variable to false gives the result.

The next lemma is used to prove in all polynomial cases $\hat{H}$ is a caterpillar. Although this section is restricted to reflexive graphs, we will prove it in greater generality for future use in a later section. To that end let $F_2$ be the graph in Figure 8 where each loop on the three leaves may or may not be present. Thus, $F_2$ represents a family of graphs, but we will abuse notation and simply refer to $F_2$ as any member of that family.
a) \[ \begin{array}{c}
1 \\
2 \\
\end{array} \]

b) \[ \begin{array}{c}
1 \\
2 \\
\end{array} \]

c) \[ \begin{array}{c}
1 \\
2 \\
\end{array} \]

\begin{itemize}
  
  \item [d)] \[ \begin{array}{c}
1 \\
2 \\
3 \\
\end{array} \]

  \[ U = 2 - 1 - 2 - 2 \\
     D = 2 - 2 - 3 - 2 \]

  \item [e)] \[ \begin{array}{c}
1 \\
2 \\
\cdots \\
k \\
\end{array} \]

  \[ \text{blue path} \]

  \[ U = 1 - 2 - \cdots - k - k \\
     D = 1 - 1 - 2 - \cdots - (k - 1) - k \]

  \item [f)] \[ \begin{array}{c}
1 \\
2 \\
\cdots \\
k - 1 \\
k \\
\end{array} \]

  \[ \text{blue path} \]

  \[ U = 1 - 2 - \cdots - (k - 1) - k - (k - 1) \\
     D = 1 - 1 - 1 - 2 - \cdots - (k - 1) - k \]

  \item [g)] \[ \begin{array}{c}
1 \\
2 \\
3 \\
\cdots \\
k - 1 \\
k \\
\end{array} \]

  \[ \text{blue path} \]

  \[ U = 2 - 3 - \cdots - k - (k - 1) \\
     D = 2 - 1 - 2 - \cdots - (k - 1) \]

  \item [h)] \[ \begin{array}{c}
1 \\
2 \\
3 \\
\cdots \\
k - 1 \\
k \\
\end{array} \]

  \[ \text{bicolored path} \]

  \[ U = 2 - 1 - 1 - 2 - \cdots - (k - 1) \\
     D = 2 - 3 - \cdots - k - k - (k - 1) \]

  \item [i)] \[ \begin{array}{c}
1 \\
2 \\
3 \\
4 \\
5 \\
\end{array} \]

  \[ U = 3 - 2 - 1 - 2 - 3 - 3 \\
     D = 3 - 3 - 4 - 5 - 4 - 3 \]

  \item [j)] \[ \begin{array}{c}
1 \\
2 \\
3 \\
4 \\
5 \\
\end{array} \]

  \[ U = 6 - 3 - 2 - 1 - 2 - 3 - 6 - 6 \\
     D = 6 - 6 - 3 - 4 - 5 - 4 - 3 - 6 \]

  \item [k)] \[ \begin{array}{c}
1 \\
2 \\
3 \\
4 \\
5 \\
\end{array} \]

  \[ U = 3 - 2 - 1 - 2 - 3 - 6 - 6 - 3 \\
     D = 3 - 6 - 6 - 3 - 4 - 5 - 4 - 5 - 4 - 3 \]

  \item [l)] \[ \begin{array}{c}
1 \\
2 \\
3 \\
4 \\
5 \\
\end{array} \]

  \[ U = 3 - 6 - 6 - 3 - 4 - 5 - 4 - 3 \\
     D = 3 - 2 - 1 - 2 - 3 - 6 - 6 - 3 \]

  \item [m)] \[ \begin{array}{c}
1 \\
2 \\
3 \\
4 \\
\end{array} \]

\end{itemize}

**Figure 6** A family $G$ of signed graphs yielding NP-complete problems. (The solid edges/loops are blue, the dashed edges/loops are red. The dotted loops can be either blue, red or bicoloured.) Some chains are indicated on the right. For e) we have $k \geq 2$, for f) $k \geq 3$, and for g) and h) $k \geq 4$.

**Figure 7** The gadget $(T, \sigma)$ for the case m.)
List homomorphism problems for signed graphs

Figure 8 The subgraph $F_2$.

Lemma 16. If the underlying graph $H$ contains the graph $F_2$ in Figure 8 then the problem $\text{List-S-Hom}(H)$ is NP-complete.

Proof. Deciding if there exists a list homomorphism (of an unsigned graph) to the graph $F_2$ is NP-complete [11]. A direct reduction of $\text{List-Hom}(F_2)$ to $\text{List-S-Hom}(H)$ as in the proof of Lemma 9 is complicated by the fact that the loops in $H$ can be red, blue, or bicoloured. However, the proof from [11] can be adapted to our setting as we now describe.

Suppose that $\hat{F}_2$ is a subgraph of $H$ with underlying graph $F_2$, and suppose that $\hat{F}_2$ has been switched so that all non-loop edges are at least blue. Label the leaves of $\hat{F}_2$ by 0, 1, 2, and their respective neighbours as $0^+, 1^+, 2^+$, and label the center vertex $c$.

If all the unicoloured loops in $\hat{F}_2$ are blue, then we may restrict the input to blue (there is no advantage to switching). The NP-complete problem $\text{List-Hom}(F_2)$ [11] reduces to $\text{List-S-Hom}(H)$. Thus, there are bicoloured loops in blue and red.

We first prove that if some edge $ci^+$, $i \in \{0, 1, 2\}$ is not bicoloured, then $\text{List-S-Hom}(\hat{F}_2)$ is NP-complete by showing that the copy of $\hat{F}_2$ contains a member of the family $G$.

Note, any path between a blue loop and a red loop must have a vertex with a bicoloured edge. Thus, at least one of $c, 0^+, 1^+, 2^+$ has a bicoloured loop.

Next, if none of the edges $ci^+$ are bicoloured, then we either have a copy of (e) from family $G$, when at least two of $c, 0^+, 1^+, 2^+$ have bicoloured loops, or a copy of either (j) or (k) from the proof of Lemma 9.

Suppose that $\hat{F}_2$ is a subgraph of $H$ with underlying graph $F_2$, and suppose that $\hat{F}_2$ has been switched so that all non-loop edges are at least blue. Label the leaves of $\hat{F}_2$ by 0, 1, 2, and their respective neighbours as $0^+, 1^+, 2^+$, and label the center vertex $c$.

If all the unicoloured loops in $\hat{F}_2$ are blue, then we may restrict the input to blue (there is no advantage to switching). The NP-complete problem $\text{List-Hom}(F_2)$ [11] reduces to $\text{List-S-Hom}(H)$. Thus, there are bicoloured loops in blue and red.

We first prove that if some edge $ci^+$, $i \in \{0, 1, 2\}$ is not bicoloured, then $\text{List-S-Hom}(\hat{F}_2)$ is NP-complete by showing that the copy of $\hat{F}_2$ contains a member of the family $G$.

Note, any path between a blue loop and a red loop must have a vertex with a bicoloured edge. Thus, at least one of $c, 0^+, 1^+, 2^+$ has a bicoloured loop.

Next, if none of the edges $ci^+$ are bicoloured, then we either have a copy of (e) from family $G$, when at least two of $c, 0^+, 1^+, 2^+$ have bicoloured loops, or a copy of either (j) or (k) from family $G$. (The chain for (k) only requires two unicoloured edges and one bicoloured edge incident at 3 with the required loop in $L$ are arbitrary.) Finally if two of the edges are bicoloured, then we have a copy of (l) from family $G$. (The edges 12 and 45 can be arbitrary.) Thus, all edge $ci^+$ are bicoloured.

We now finish the proof using a modification of the proof in [11]. Given distinct $i$ and $j$ in $\{0, 1, 2\}$ and distinct subsets $I$ and $J$ of $\{0, 1, 2\}$, an $(i, I, j, J)$-chooser is a path $\hat{P}$ with endpoints $a$ and $b$, together with a list assignment $L$, such that the following statement holds. For each list homomorphism $f$ from $\hat{P}$ to $\hat{F}_2$, either $f(a) = i$ and $f(b) \in I$ or $f(a) = j$ and $f(b) \in J$. Moreover, for each $i' \in I$ and $j' \in J$, there are list homomorphisms $g_1, g_2$ from $\hat{P}$ to $\hat{F}_2$ such that $g_1(a) = i$, $g_1(b) = i'$ and $g_2(a) = j$, $g_2(b) = j'$.

Suppose $\hat{P}$ is a $(0, \{0, 1\}, 1, \{1, 2\})$-chooser, $\hat{P}'$ is $(0, \{1, 2\}, 1, \{2, 0\})$-chooser, and $\hat{P}''$ is a $(0, \{2, 0\}, 1, \{0, 1\})$-chooser. Let $\hat{T}$ be the tree obtained by identifying the $b$ vertices in the three choosers and labelling the leaves respectively as $a, a', a''$. It is easy to verify that $\hat{T}$ admits a list-homomorphism to $\hat{F}_2$ if, and only if, the triple $(a, a', a'')$ does not map to either $(0, 0, 0)$ or $(1, 1, 1)$. Consequently, we can reduce an instance of NOT-ALL-EQUAL SAT to $\text{List-S-Hom}(\hat{F}_2)$. For each clause in the instance, create a copy of $\hat{T}$ and identify the vertices $(a, a', a'')$ with the three literals in the clause.

It remains to construct the choosers. First, we build a $(0, \{0, 2\}, 1, \{1, 2\})$-chooser. By symmetry we then have $(i, \{i, k\}, j, \{j, k\})$-choosers for any distinct $i, j, k \in \{0, 1, 2\}$. Let $Q$
be a path on \( q_0, q_1, \ldots, q_{10} \) with lists

\[
\begin{align*}
L(q_0) &= \{0, 1\} & L(q_6) &= \{0^+, 2^+, 1\} \\
L(q_1) &= \{0^+, 1^+\} & L(q_7) &= \{0^+, c, 1^+\} \\
L(q_2) &= \{0, 1^+\} & L(q_8) &= \{0, 2^+, 1\} \\
L(q_3) &= \{0^+, c, 1^+\} & L(q_9) &= \{0^+, 2^+, 1^+\} \\
L(q_4) &= \{0, 2^+, 1\} & L(q_{10}) &= \{0, 2, 1\} \\
L(q_5) &= \{0^+, 2, 1^+\}
\end{align*}
\]

The path \( \hat{Q} \) has all edges blue. In mapping \( \hat{Q} \) to \( \hat{F}_2 \) first suppose \( q_0 \) maps to 0. Then \( q_{10} \)
either maps to 0, in which case the loop \( 0^+ \) is traversed twice, or \( q_{10} \) maps to 2 in which case the loop at \( 0^+ \), then edge \( 0^+ c \) and the loop at \( 2^+ \) are each traversed once. In the former case the image of the path is positive. In the latter case, since \( 0^+ c \) and \( c 2^+ \) are bicoloured, there is a positive walk from 0 to 2 to which \( \hat{Q} \) maps. Note if there is a red loop at \( 0^+ \) we switch at \( q_6 \). If there is a red loop at \( 2^+ \) and \( q_7 \) maps to \( c \), then we switch at \( q_8 \). Similarly reasoning shows \( \hat{Q} \) can map to \( \hat{F}_2 \) with \( q_0 \) mapping to 1 and \( q_{10} \) mapping to either 1 or 2 but not to 0. Thus \( \hat{Q} \) is a \( (0, \{0, 2\}, 1, \{1, 2\}) \)-chooser.

The \( (0, \{0\}, 1, \{2\}) \)-chooser \( \hat{R} \) is a path with vertices \( r_0, \ldots, r_6 \) and lists

\[
\{0, 1\}, \{0^+, 1^+\}, \{0, 1^+\}, \{0^+, c\}, \{0, 2^+\}, \{0^+, 2^+\}, \{0, 2\}.
\]

All edges are blue. When \( \hat{R} \) maps to the edge \( 00^+ \), the image is positive. When \( \hat{R} \) maps to the path \( 1, 1^+, 1^+, c, 2^+, 2^+, 2 \), switching at \( r_2 \) or \( r_4 \) is required when there is a red loop at \( 1^+ \) or \( 2^+ \) respectively.

The required choosers are defined as follows. First, \( \hat{P} \) is the \( (0, \{0\}, 1, \{2\}) \)-chooser followed by the \( (0, \{0, 1\}, 2, \{1, 2\}) \)-chooser. Next \( \hat{P}' \) is the concatenation of the \( (0, \{0\}, 1, \{2\}) \)-chooser, the \( (0, \{1\}, 2, \{2\}) \)-chooser, the \( (1, \{1\}, 2, \{0\}) \)-chooser, and the \( (1, \{1, 2\}, 0, \{0, 2\}) \)-chooser. Finally \( \hat{P}'' \) is the concatenation of the \( (0, \{2\}, 1, \{1\}) \)-chooser and the \( (2, \{0, 2\}, 1, \{0, 1\}) \)-chooser.

A reflexive tree \( H \) is a **caterpillar** if it contains a path \( P = v_1 \ldots v_k \) such that each vertex of \( H \) is on \( P \) or is adjacent to a vertex of \( P \). Note that the path \( P \), which we again call the **spine** of \( H \), is not unique, and we sometimes make it explicit by saying that \( H \) is a caterpillar **with spine** \( P \). A vertex \( x \) not on \( P \) is adjacent to a unique neighbour \( v_i \) on \( P \), and we call the edge \( v_i x \) (with the loop at \( x \)) the **subtree rooted at** \( v_i \). A vertex on the spine can have more than one subtree rooted at it. We say that \( \hat{H} \) is a **good caterpillar with respect to the spine** \( v_1 \ldots v_k \) if the bicoloured edges of \( \hat{H} \) form a connected subgraph, the unicoloured non-loop edges all have the same colour \( c \), and there exists an integer \( d \), with \( 1 \leq d \leq k \), such that

- all edges on the path \( v_1 v_2 \ldots v_d \) are bicoloured, and all edges on the path \( v_d v_{d+1} \ldots v_k \) are unicoloured with colour \( c \),
- all loops at the vertices \( v_1, \ldots, v_{d-1} \) and all non-loop edges of the subtrees rooted at these vertices are bicoloured,
- all loops at the vertices \( v_{d+1}, \ldots, v_k \) and all edges and loops of the subtrees rooted at these vertices are unicoloured with colour \( c \),
- if \( v_d \) has a bicoloured loop, then all children of \( v_d \) with bicoloured loops are adjacent to \( v_d \) by bicoloured edges,
- if \( v_d \) has a unicoloured loop of colour \( c \), then all children of \( v_d \) have unicoloured loops of colour \( c \), and are adjacent to \( v_d \) by unicoloured edges, and
- if \( d < k \), then the loops of all children of \( v_d \) adjacent to \( v_d \) by unicoloured edges also have colour \( c \).
The vertex $v_d$ will again be called the \textit{dividing vertex}. We also say that $\hat{H}$ is a \textit{good caterpillar} \textit{with preferred colour $c$}. Figure \ref{fig:good-caterpillar} (on the left) shows an example of good caterpillar with preferred colour blue. We emphasize that in the case $d = k$ (depicted in Figure \ref{fig:good-caterpillar} on the right), it is possible (if $v_d$ has a bicoloured loop) that $v_d$ has some children with red loops and some with blue loops, adjacent to $v_d$ by unicoloured edges.

Let $G$ be the family of signed graphs depicted in Figure \ref{fig:good-caterpillar} together with the family of complementary signed graphs where all unicoloured edges and loops are red, rather than blue, and vice versa. Note that the complementary signed graphs are not switching equivalent to the original signed graphs because switching does not change the colour of loops.

\begin{lemma}
Let $\hat{H}$ be a reflexive signed tree. Then $\hat{H}$ is a good caterpillar if and only if it does not contain any of the graphs in the family $G$ as an induced subgraph, and the underlying graph $H$ does not contain the graph $F_2$.
\end{lemma}

\begin{proof}
It is easy to see that none of the signed reflexive trees in Figure \ref{fig:good-caterpillar} is a good caterpillar. By symmetry, the same is true for their complementary signed graphs. It is also clear that the graph $F_2$ (from Figure \ref{fig:good-caterpillar}) is not a caterpillar. We proceed to show that if the signed reflexive trees from family $G$ in Figure \ref{fig:good-caterpillar} are excluded as induced subgraphs, then $\hat{H}$ is a good caterpillar with preferred colour blue. (The complementary exclusions produce a good caterpillar with preferred colour red.) Since the graphs g) are absent, the bicoloured non-loop edges induce a connected subgraph. The exclusion of families h) and l) ensures there is a spine $P = v_1 \ldots v_k$, and a dividing vertex $v_d$, such that all edges between $v_1, \ldots, v_{d-1}$ and their children are bicoloured. The exclusion of b) and c) ensures each bicoloured non-loop edge has a bicoloured loop on (at least) one of its endpoints. Forbidding the family d) ensures the vertices $v_1, \ldots, v_{d-1}$ all have bicoloured loops.

The subgraph induced by $v_{d+1}, \ldots, v_k$, and its leaves must only contain blue edges (and loops) since the edge $v_d v_{d+1}$ is blue and the bicoloured edges induce a connected subgraph. (Recall that when we say blue we mean unicoloured blue.) Forbidding a), e), and f) implies all the loops in this subgraph are blue. (In the case $v_1 = v_d$, if there is a bicoloured loop on exactly one leaf of $v_1$, we make that leaf the first vertex of the spine.)

Excluding family e) ensures a leaf of $v_d$ with a bicoloured loop must join $v_d$ with a bicoloured edge, provided $d > 1$. If $d = 1$ and $v_1$ has multiple leaves with bicoloured loops, then cases d) and e) ensure $v_1$ has a bicoloured loop and all edges from the bicoloured loop leaves to $v_1$ are bicoloured. In the case there is a unique leaf with a bicoloured loop, we make that leaf the first vertex of the spine.

Finally if $d < k$, case m) implies that we can choose the spine so that no leaf child of $v_d$ has a red loop.

Families i), j), and k) ensure when there is a single bicoloured loop or a single bicoloured non-loop edge, the spine begins with this loop or edge.
\end{proof}

\begin{theorem}
Let $\hat{H}$ be a reflexive tree. If $\hat{H}$ is a good caterpillar, then the problem $\text{LIST-S-HOM}(\hat{H})$ is polynomial-time solvable. Otherwise, $H$ contains $F_2$ from Figure \ref{fig:good-caterpillar} or
\end{theorem}
Suppose that $\hat{H}$ is not a good caterpillar. If $H$ is not a caterpillar, then it contains $F_2$ from Figure 3, and the problem is NP-complete. Otherwise, $\hat{H}$ contains an induced subgraph from $\mathcal{G}$, and the problem is NP-complete by Lemma 15.

We prove the first statement. Thus assume that $\hat{H}$ is a good caterpillar, with spine $v_1 \ldots v_k$ and dividing vertex $v_d$. By symmetry, we may assume it is a good caterpillar with preferred colour blue. We distinguish three types of rooted subtrees.

- **Type $T_1$:** a bicoloured edge $v_ix$ with a bicoloured loop on $x$;
- **Type $T_2$:** a bicoloured edge $v_ix$ with a unicoloured loop on $x$;
- **Type $T_3$:** a blue edge $v_ix$ with a unicoloured loop on $x$.

There is a general version of min ordering we can use in this context. A **min ordering** of a graph $H$ is a linear ordering $< \subseteq V(\hat{H})$ such that for vertices $x < x', y < y'$, if $xy', x'y$ are both edges in $H$, then $xy$ is also an edge in $H$. It is again the case that if a graph $H$ admits a min ordering, then the list homomorphism problem for $H$ can be solved in polynomial time by arc consistency followed by making the minimum choice in each list [13].

Supposing again that $\hat{H}$ is a good caterpillar with spine $v_1 \ldots v_k$ and preferred colour blue. A **special min ordering** of $\hat{H}$ is a min ordering of the underlying graph $H$ such that for any vertices $v_i, x, x'$ with edges $v_ix, v_ix'$ we have $x < x'$ if

- the edge $v_ix$ is bicoloured and the edge $v_ix'$ is blue, or
- $x$ has a bicoloured loop and $x'$ a unicoloured loop, or
- $x$ has a blue loop and $x'$ has a red loop.

**Lemma 19.** Every good caterpillar $\hat{H}$ admits a special min ordering.

**Proof.** It is again easy to see that the ordering $v_1 < v_2 < \ldots < v_k$ of $V(\hat{H})$ in which the children of each $v_i$ are ordered between $v_i$ and $v_{i+1}$ is a min ordering of the underlying graph $H$. We may again assume that $\hat{H}$ has preferred colour blue. To ensure that $<$ is a special min ordering of $\hat{H}$, we make sure that after each vertex $v_i$ with $i = 1, 2, \ldots, d - 1$, we first list the leaves of subtrees of type $T_1$, then the leaves of subtrees of type $T_2$ with blue loop, and last the leaves of subtrees of type $T_2$ with red loop. If $d = k$, then we proceed the same way also after $v_d$, and then we list the leaves of subtrees of type $T_3$ with blue loop, and last the leaves of subtrees of type $T_3$ with red loop. If $d < k$, we list after $v_d$ first the leaves of subtrees of type $T_1$, then the leaves of subtrees of type $T_2$ with blue loop, then the leaves of subtrees of type $T_2$ with red loop, and last the leaves of subtrees of type $T_3$. For vertices $v_i, i > d$, there are only subtrees of type $T_3$, and their leaves can be listed in any order. ▶

We now describe our polynomial-time algorithm. As in the irreflexive case, we first perform the arc consistency test to check for the existence of a homomorphism of the underlying graphs ($G$ to $H$). Then we also perform the bicoloured arc consistency test. If we obtain an empty list, there is no list homomorphism. Otherwise, taking again the minima of all lists (in the special min ordering $<$) defines a list homomorphism $f : G \to H$ of the underlying graphs by [13], and again by bicoloured arc consistency test we have that $f$ maps bicoloured edges of $G$ to bicoloured edges of $\hat{H}$. Therefore, by Lemma 19 and the remarks following it, $f$ is also a list homomorphism of the signed graphs $\hat{G} \to \hat{H}$, unless a negative cycle $C$ of unicoloured edges of $\hat{G}$ maps to a positive closed walk $f(C)$ of unicoloured edges in $\hat{H}$, or a positive cycle $C$ of unicoloured edges of $\hat{G}$ maps to a negative closed walk $f(C)$ of unicoloured edges in $\hat{H}$. The minimum choices in all lists imply that no vertex $x$ of $C$ can be mapped to an image $y$ with $y < f(x)$. We proceed to modify the images of such cycles $C$ one
by one, in the order of increasing smallest vertex in $f(C)$ (in the ordering $<$), until we either obtain a homomorphism of signed graphs, or we find that no such homomorphism exists.

Let $w$ be the leaf of the last subtree of type $T_2$ rooted at $v_d$. We note that if $d < k$, then all edges and loops amongst the vertices that follow $w$ in $<$ are blue, by the properties of a special min ordering. We distinguish three possible cases.

- **At least one vertex $y$ of $f(C)$ satisfies $y < w$:***
  
  The only unicoloured closed walks including $y$ are (red or blue) loops, so $f$ maps the entire cycle $C$ to $y$. As in the reflexive case, we may remove $y$ from all lists of vertices of $C$ and continue seeking a better homomorphism of the underlying graphs $(G$ to $H)$.

- **All vertices of $f(C)$ follow $w$ in the order $<$ and $d < k$:***
  
  In this case $C$ is a negative cycle of unicoloured edges. The subgraph of $\hat{H}$ induced by vertices after $w$ in the order $<$ has only blue edges and loops. Thus there is no homomorphism of signed graphs mapping $\hat{G} \to \hat{H}$.

- **All vertices of $f(C)$ follow $w$ in the order $<$ and $d = k$:***
  
  In this case a fairly complex situation may arise because $f(C)$ can be a closed walk using both red and blue loops, along with blue edges; see below.

We now consider the final case in detail. Since $f$ chooses minimum possible values of images (under $<$), we could only modify $f$ by mapping some vertices of $C$ that were taken by $f$ to a vertex with a blue loop, to vertex with a red loop instead, if lists allow it. We show how to reduce this problem to solving a system of linear equations modulo two, which can then be solved in polynomial time by (say) Gaussian elimination. We begin by considering the pre-image (under $f$) of all vertices in the subtrees of type $T_3$ rooted at $v_d$. We denote by $P$ the set of vertices $v \in V(G)$ with $f(v)$ equal to a vertex with a blue loop and by $N$ the set of vertices $v \in V(G)$ with $f(v)$ equal to a vertex with a red loop. We say that a vertex $x$ of $G$ is a boundary point if $f(x) = v_d$. The set of boundary points is denoted by $B$. Thus the pre-image of the subtrees of type $T_3$ rooted at $v_d$ is the disjoint union $B \cup P \cup N$. We now focus on the subgraph $\hat{G}'$ of $\hat{G}$ induced by $B \cup P \cup N$. A region is a connected component of $\hat{G}' \setminus B$ together with all its boundary points, i.e. between any pair of vertices in a region there is a path with no boundary point as an internal vertex.

Given a region $r$ and boundary points $x$ and $y$ (not necessarily distinct), we construct (possibly several) boolean equations on the corresponding variables, using the same symbols $x, y$, and $r$. The variables $x,y$ indicate whether or not the corresponding boundary vertices $x$ and $y$ should be switched before mapping them with $f$ (true corresponds to switching), and the variable $r$ indicates whether the region $r$ will be mapped by $f$ to a blue loop or a red loop (true corresponds to a blue loop). The equations depend of the parity and the sign of walks between the two vertices. If $c$ and $d$ denote parities (even or odd), we say a walk $W$ from $x$ to $y$ in $\hat{G}'$ is a $(c,d)$-walk if it contains no boundary points other than $x$ and $y$, the parity of the number of blue edges in $W$ is $c$, and the parity of the number of red edges in $W$ is $d$. The equations generated by the $(c,d)$-walks are as follows.

- **(odd,odd)-walk:** We add the equation $x = y + 1$. This ensures that exactly one of the boundary vertices has to be switched, in particular $x$ and $y$ must be distinct. The image of the walk must be uni-balanced or anti-uni-balanced (as the whole walk maps to exactly one subtree of type $T_3$). An even length walk with an odd number of red edges is neither. However, if we switch at exactly one of the endpoints, we can freely map all of the non-boundary points to a blue loop or a red loop.

- **(even,even)-walk:** We add the equation $x = y$. The reasoning is similar to the previous case.

- **(odd,even)-walk:** We add the equation $x = y + r + 1$. The image of the walk is a closed
walk of odd length and positive sign. Thus if both or neither of $x$ and $y$ are switched, then the walk remains positive and $r = 1$. Conversely, switching exactly one of $x$ or $y$ makes the walk negative, and $r = 0$.

- \((even, odd)\)-walk: We add the equation $x = y + r$. The argument is analogous to the previous case.

It is possible that there are several kinds of walks between the same $x, y$, but we only need to list one of each kind, so the number of equations is polynomial in the size of $G$. A simple labelling procedure can be used for determining which kinds of walks exist, for given boundary points $x$ and $y$ and a region $r$. We start at the vertex $x$, and label its neighbours $n_x$ by the appropriate pairs $(c, d)$, determined by the signs of the edges $xn_x$. Once a vertex is labelled by a pair $(c, d)$, we correspondingly label its neighbours; a vertex is only given a label $(c, d)$ once even if it is reached with that label several times. Thus a vertex has at most four labels. Any time a vertex receives a new label its neighbours are checked again.

The process ends in polynomial time (in the size of the region) as each edge of the region is traversed at most four times. The result is inherent in the labels obtained by $y$.

Finally, for each region we examine the connected component of the non-boundary vertices. Since the arc consistency procedure was done in the first step of the algorithm, all lists of non-boundary points for a given region are the same. Also, by the ordering $<$, these lists must only contain leaves of $v_d$. Thus, the non-boundary vertices of the region must map to a single loop. We ensure the choice of the loop is consistent with the lists of each region. If the lists of vertices of some region do not contain a vertex with a red loop, then we add the equation $r = 1$ for the region. Similarly, if the lists do not contain a blue loop, then we add the equation $r = 0$.

Such a system of boolean linear equations can be solved in polynomial time. Also, the system itself is of polynomial size measured by the size of $\hat{G}$. This completes the proof.

7 General trees

In this section we handle signed trees $\hat{H}$ in general, i.e., trees in which some vertices have loops and others don’t. There is a certain amount of duplication with the previous cases, but we have kept the simpler cases separate since some of those results are used here. To simplify the descriptions, we assume, without loss of generality, that all non-loop unicoloured edges are blue, unless noted otherwise.

In Figure 10 we introduce our main NP-complete cases.

We first focus on signed trees $\hat{H}$ without bicoloured non-loop edges. If there are no bicoloured loops either, then Theorem 6 implies that List-S-Hom($\hat{H}$) is NP-complete when $\hat{H}$ has both a red loop and a blue loop, or when the underlying graph is not a bi-arc tree. We now introduce NP-complete cases when bicoloured loops are allowed.

- \textbf{Lemma 20.} If $\hat{H}$ contains any of the graphs a) - d) in the family $J$ in Figure 10, then the problem List-S-Hom($\hat{H}$) is NP-complete.

\textbf{Proof.} For each of the signed graphs a), b) and c) in family $J$ we can apply Theorem 8. The figure lists a chain for each of these forbidden subgraphs.

In the final case d), we reduce Not-All-Equal SAT to List-S-Hom($H, \pi$) where $(H, \pi)$ is the signed graph d) in family $J$. Let $(T', \sigma')$ be the signed graph with the list assignments and signature shown in Figure 11. For each clause $(x, y, z)$ in the instance of Not-All-Equal SAT, we create a copy of $(T', \sigma')$ identifying the leaves $x, y, z$ in $T'$ with the variables in the clause.
List homomorphism problems for signed graphs

![Images of graphs a) to r)](figure)

**Figure 10** The family \( \mathcal{F} \). (The dotted loops can be arbitrary or missing, unless stated otherwise.)
We claim that \((T', \sigma')\) has a list homomorphism to \((H, \pi)\) if and only if we switch at exactly one or two elements of \(\{x, y, z\}\). We can then view the switching at one of \(\{x, y, z\}\) as setting the variable to true and, conversely, no switching as setting to false.

Consider a mapping of \((T', \sigma')\) to \((H, \pi)\). It is easy to see that either both \(x\) and \(m\) are switched or neither is switched. We also observe that if \(m\) maps to 1, then exactly one of \(m\) or \(y\) must be switched. On the other hand, if \(m\) maps to 3, then neither or both of \(m\) and \(y\) is switched. (In the first case the image of the path is a negative walk, while in the second case it is a positive walk.) Thus, when \(m\) maps to 1, exactly one of \(x\) or \(y\) is switched, and when \(m\) maps to 3, either both or neither \(x\) and \(y\) is switched. Finally, if \(m\) maps to 1, then we are free to switch or not switch at \(z\). Contrary to that, if \(m\) maps to 3, then we must switch at \(z\) if and only if we do not switch at \(m\).

If bicoloured edges are present, we use the following result. Note that some of the cases and chains are the same as for reflexive graphs or irreflexive graphs.

\begin{lemma}
If \(\hat{H}\) contains any of the graphs e) - n) in family \(\mathcal{J}\) in Figure 10, then the problem \(\text{List-S-Hom}(\hat{H})\) is NP-complete.
\end{lemma}

\textbf{Proof.} For each of the signed graphs e) - n) in family \(\mathcal{J}\) we can apply Theorem \ref{thm:NP-completeness}. The figure lists a chain for each of these forbidden subgraphs, except the case j), which follows from a result in \cite{12} implying that the problem is NP-complete if the vertices with loops of any colour are disconnected. Thus any signed graph \(\hat{H}\) that contains one of the signed graphs in the cases a) - n) of the family \(\mathcal{J}\) as an induced subgraph has the problem \(\text{List-S-Hom}(\hat{H})\) NP-complete.

For future reference we note that we can also use the signed trees in family \(\mathcal{G}\), which yield NP-complete problems by Lemma \ref{lem:NP-completeness} in fact, they remain NP-complete if we omit the loop at 2 in b) or c) or d), or omit any loops at vertices 2 \(\ldots\) \(k - 1\) in e), or omit any loops at 2 \(\ldots\) \(k\) in f), or omit any loops at 1 \(\ldots\) \(k\) in g). We can also omit any loops at 3, \(\ldots\), \(k - 2\) in h), as long as 1 has any loop or 2 has a blue loop, and \(k\) has any loop or \(k - 1\) has a blue loop. In addition, all loops can be arbitrary or missing if instead of the blue edge 12 there is a blue path 012 and instead of the blue edge \((k - 1)k\) there is a blue path \((k - 1)k + 1\). The remaining trees of family \(\mathcal{G}\) can also be used, with the loops at 1, 2, 4, 5 arbitrary or missing in i), loops at 1, 2, 3, 4, 5 arbitrary or missing in j), all loops arbitrary or missing, except an arbitrary loop present at 6 in k) and l), and the loop at 4 missing in m). The proofs of these facts follow from the corresponding chains in Figure 6 or the chains in Figure 10.

In cases p), q), and r) in Figure 10 we present three additional NP-complete trees we will use. Note that the case q) is a special case of p), with \(k = 2\). The case q) is also a special case of a) in family \(\mathcal{J}\), where the chain is specified in general. However, the chain for the case p) when \(k > 2\) is different, as shown in the figure. (Note that the absence of a loop at 2 is crucial for the chain in p), and for r), the absence of a loop at 4 is crucial, while the edges 12 or 45 could be blue or bicoloured and the given chain would still apply.)

Thus we have the following lemma.
Lemma 22. If \( \hat{H} \) contains any of the graphs p), q), r) in family \( J \) in Figure [10], then the problem \( \text{List-S-Hom}(\hat{H}) \) is NP-complete.

If \( \hat{H} \) is a signed graph, the \textit{bicoloured part} of \( \hat{H} \) is the graph \( D\hat{H} \) (with possible loops) consisting of all those edges and loops that occur as bicoloured edges and loops in \( \hat{H} \), and all the vertices they contain. (Thus vertices of \( \hat{H} \) not incident with a bicoloured edge or loop are deleted.) Similarly, the \textit{blue part} of \( \hat{H} \) is the graph \( B\hat{H} \) with possible loops consisting of all those edges (and loops) that are at least blue in \( \hat{H} \). Since we assume all non-loop edges of \( \hat{H} \) are blue, every vertex of \( \hat{H} \) is included in \( B\hat{H} \). (We may think of the letter \( B \) as standing for "blue" and the letter \( D \) as standing for "double", in the sense of having both colours.)

We now denote by \( T \) the union of all the NP-complete tree families \( \mathcal{G}, \mathcal{H}, \mathcal{J} \). There are further cases that cause the problem to be NP-complete. Theorem 6 implies, in the context of trees, that the problem is NP-complete if there are no bicoloured edges or loops and there is both a red loop and a blue loop. Any signed graph \( \hat{H} \) which is not irreflexive and has a bicoloured edge but no bicoloured loops yields an NP-complete homomorphism (and hence list homomorphism) problem by Theorem 5 since the s-core contains at least one unicoloured loop and one bicoloured edge (counted as two edges). As discussed earlier, if the vertices with loops of any fixed colour induce a disconnected graph, the problem is NP-complete by [12]. Finally, as mentioned earlier, if the bicoloured part \( D\hat{H} \) yields an NP-complete list homomorphism problem, then so does \( \hat{H} \), since for bicoloured inputs, this is the only part of \( \hat{H} \) that can be used. Thus \( \text{List-S-Hom}(\hat{H}) \) is also NP-complete if the unsigned graph \( D\hat{H} \) is not a bi-arc tree, i.e., contains one of the trees in Figures 3 and 4 of [11]. Moreover, if \( \hat{H} \) contains no red loops, then it is also true that if the blue part \( B\hat{H} \) yields an NP-complete list homomorphism problem, then so does \( \hat{H} \). Indeed, if there are no red loops (or edges) in \( \hat{H} \), then for an input signed graph \( \hat{G} \) that has only blue edges, there is no cause for switching. In other words a blue input \( \hat{G} \) admits a signed list homomorphism to \( \hat{H} \) if and only if \( G \) admits a list edge-coloured homomorphism to \( H \). This is a reduction from the list homomorphism problem for \( B\hat{H} \) to the signed list homomorphism problem for \( \hat{H} \).

We say that a signed tree is \textit{colour-connected} if each of the following subgraphs is connected: the subgraph spanned by edges that are at least blue, the subgraph spanned by edges that are at least red, the subgraph spanned by edges that are bicoloured, the subgraph induced by the vertices with loops that are at least blue, the subgraph induced by the vertices with loops that are at least red, and the subgraph induced by the vertices with loops that are bicoloured.

We call a signed tree \( \hat{H} \) a \textit{good signed tree} if it satisfies the following conditions.

1. If \( \hat{H} \) has no bicoloured edge, then all the loops are of the same colour (red or blue).
2. If \( \hat{H} \) has a bicoloured non-loop edge, then it also has a bicoloured loop, or it has no loops at all.
3. \( \hat{H} \) is colour-connected.
4. The blue part \( B\hat{H} \) is a bi-arc tree.
5. \( \hat{H} \) contains no signed tree from the family \( T \).

7.1 Assuming no red loops

In this subsection, we assume that \( \hat{H} \) has no red loops. It follows from the previous section, that if such \( \hat{H} \) is not good, then \( \text{List-S-Hom}(\hat{H}) \) is NP-complete. In the next subsection, we prove this is true for signed trees with red loops as well. In this subsection, we analyse the structure of good signed trees without red loops.
Proposition 23. Let $\tilde{H}$ be a good signed tree without red loops but with at least one bicoloured loop. Then $\tilde{H}$ is either

- obtained from a good reflexive caterpillar (with spine $v_1, v_2, \ldots, v_k$) by removing loops at a subset $S$ of leaves, and
- optionally replacing any bicoloured edges $v_iu$ by blue edges for these leaves $u \in S$, or
- is a signed 2-caterpillar (with spine $v_1, v_2, \ldots, v_k$) obtained from a bi-arc tree by replacing each edge and loop by bicoloured edge and loop (respectively),
- optionally, for Type (b) 2-caterpillars, adding a blue loop at a leaf adjacent to $v_1$, and
- optionally adding, at a spine vertex $v_i$ or at a loopless child of a $v_i$, a blue edge leading to a new (loopless) leaf.

Proof. Let $\tilde{H}$ be a good signed tree with no red loops and at least one bicoloured loop. Since the blue part $B_{\tilde{H}}$ is a bi-arc tree, it is of one of two types described in Figures 5 and 6 of [11], repeated here for completeness as Figures 12 and 13. Namely, they are a caterpillar obtained from a reflexive bi-arc caterpillar by removing loops on some leaves, or a 2-caterpillar obtained from an irreflexive bi-arc tree by the addition of loops on some allowed vertices as described in Theorem 5.1 of [11].

Assume first that the blue part $B_{\tilde{H}}$ is a bi-arc tree of the first type, in other words, a caterpillar with loops on all vertices on the spine and possibly some leaves. We now proceed analogously to the proof of Lemma 17, using the remarks after the proof of Lemma 21 explaining how to use the trees related to those in family $G$. The only difference from the
reflexive case is caused by the requirement of loops in cases (h) and (l) from the family $G$, resulting in the fact that some vertices $v_1, v_2, \ldots, v_{d-1}$ can have incident blue edges off the spine, as long as they lead to vertices without loops, as enforced by the absence of signed trees from the family $J$.

Thus $\hat{H}$ is indeed a caterpillar obtained from a reflexive signed caterpillar by removing loops at some leaves, and optionally replacing the bicoloured edges by blue edges to some of those leaves.

It remains to consider the case when the blue part $B_{\hat{H}}$ is a 2-caterpillar obtained by adding suitable loops to an irreflexive bi-arc tree, cf. the two bi-arc trees in Figure 13. Specifically, there are two cases to consider.

In the first case, the blue part $B_{\hat{H}}$ has two loops, at least one of which is bicoloured. According to [11], we may choose the spine so that one loop of $B_{\hat{H}}$ is at $v_1$ and the other at its child $u$. If the loop at $v_1$ is bicoloured in $\hat{H}$, then the absence of $p)$ and $q)$ in family $J$ implies that we may assume the spine consists of bicoloured edges only, and if a vertex $v_i$ on the spine has a (necessarily loopless) neighbour $w$ that is not a leaf, then the edge $v_iw$ is bicoloured. The neighbour $u$ has a loop and needs to be considered separately. We first claim that the edge $uw_1$ must be bicoloured, else $\hat{H}$ contains the subtree $n)$ from the family $J$ (with 2 corresponding to $v_1$), or g) from the family $J$ (with $k = 2$). Moreover, if the loop at $u$ is unicoloured, then $u$ must be a leaf, otherwise $\hat{H}$ would contain r) from the family $J$. If the loop at $v_1$ is unicoloured, then the loop at $u$ must be bicoloured (we assumed that a bicoloured loop exists). Now, unless $\hat{H}$ arose from a reflexive caterpillar, it must contain a) from the family $J$ (if $uw_1$ is blue), or l) from family $J$ (if $uw_1$ is bicoloured). (Note that in both cases, the chain applies even if the edges 23, 34 are bicoloured.) In conclusion, in this case we either have both loops at $v_1$ and $u$ (as well as the edge joining them) bicoloured, or the loop at $v_1$ and the edge $uw_1$ is bicoloured, the loop at $u$ is blue and a leaf. The former situation is depicted on the left of Figure 15 (with $u$ depicted on the spine), and the latter situation is a special case of the tree on the right, with only one child ($u$) of $v_1$ having a (blue) loop. Thus going from $D_{\hat{H}}$ to $\hat{H}$ we only added a blue loop on a leaf $u$ adjacent to $v_1$, and then added some blue edges leading to leaves from any spine vertex $v_i$, or from any child of $v_2, v_3, \ldots, v_k$, or from any child of $v_1$ other than $u$.

In the second case, the blue part $B_{\hat{H}}$ has one loop at $v_1$ and possibly several other loops at leaf children of $v_1$. If the loop at $v_1$ is bicoloured, and possibly some of the loops at its children are also bicoloured, then the proof proceeds exactly as in the previous case, concluding that any edge joining two vertices with loops must be bicoloured (else there would be a copy of the subtree $n)$ from the family $J$) and the children of $v_1$ with loops are leaves. If, say, leaf $u$ has a bicoloured loop and all other loops, including the loop at $v_1$, are blue in $\hat{H}$, we again obtain a contradiction to the absence of a) from family $J$ or l) from family $J$, unless $\hat{H}$ arose from a reflexive caterpillar. In conclusion, in this case, going from $D_{\hat{H}}$ to $\hat{H}$ involved only the addition of blue loops on leaves adjacent to $v_1$, and a possible addition of some blue edges from spine vertices or from non-loop children of spine vertices, leading to leaves as described.

7.2 Allowing red loops

We now consider signed graphs $\hat{H}$ in which red loops are allowed. We denote by $\hat{H}'$ the signed tree obtained from $\hat{H}$ by deleting all vertices with red loops. We focus on the blue part $B_{\hat{H}'}$ instead of $B_{\hat{H}}$ because $\hat{H}'$ has no red loops and satisfies the assumptions of Proposition 13.
Proposition 24. Let $\tilde{H}$ be a good signed tree with at least one bicoloured loop.
Then $\tilde{H}$ is either

- obtained from a good reflexive caterpillar (with spine $v_1, v_2, \ldots, v_k$) by
  - removing loops at a subset $S$ of leaves, and
  - optionally replacing any bicoloured edges $v_i u$ by blue edges for these leaves $u \in S$, or
- is a signed 2-caterpillar (with spine $v_1, v_2, \ldots, v_k$) obtained from a bi-arc tree by
  - replacing each edge and loop by bicoloured edge and loop (respectively),
  - optionally, for Type (b) 2-caterpillars, adding a unicoloured loop at any leaf adjacent to $v_1$, and
  - optionally adding, at a spine vertex $v_i$ or at a loopless child of a $v_i$, a blue edge leading to a new (loopless) leaf.

Proof. Since $\tilde{H}'$ (defined above) satisfies the assumptions of Proposition 23, the tree $\tilde{H}'$ is described by the proposition, and we now consider where can the vertices of $B_{\tilde{H}} - B_{\tilde{H}'}$ be added, without violating any of the assumptions on $\tilde{H}$. Since the vertices with red loops must form a connected subgraph, they must be adjacent to each other and then to vertices with bicoloured loops. We now take in turn each case in the previous proof.

For the first case, when $\tilde{H}'$ is a caterpillar with reflexive spine vertices, adding a vertex with a red loop adjacent to a vertex on the spine, results in another good reflexive caterpillar with some loops on leaves removed. Adding a vertex with a red loop to a leaf with a red loop either creates a copy of $F_2$, or results in another good caterpillar with a different spine, and possibly different preferred colour, from which some loops at leaves have been removed.

For the second case, we may add a vertex $w$ with a red loop joined to $v_1$ by a bicoloured edge. If we tried to add a vertex $w$ with a red loop adjacent to a child $u$ of $v_1$, then both $wv_1$ and $wu$ would need to be bicoloured, and $u$ would have to have a bicoloured loop or a red loop. In this case, the signed tree either is a caterpillar with reflexive spine and we are in the previous case, or we would obtain, in red, a path with three loops and two non-loops.

In both cases of the proof above, we note that when the red loops are deleted (without deleting their vertices), we obtain a signed graph which also satisfies the assumptions of Proposition 23. Moreover, if the red loops are all changed to be blue, the same conclusion holds. These observations justify the following corollary.
Corollary 25. Suppose $\hat{H}$ is a signed tree. If the blue part $B_{\hat{H}}$ is not a bi-arc tree, then $\text{LIST-S-HOM}(\hat{H})$ is NP-complete. If the underlying unsigned tree is not a bi-arc tree, then $\text{LIST-S-HOM}(\hat{H})$ is NP-complete.

It follows from the first statement of Corollary 25 that if a signed tree is not good then $\text{LIST-S-HOM}(\hat{H})$ is NP-complete even if there are red loops in $\hat{H}$.

We now state our main theorem of this section.

Theorem 26. If $\hat{H}$ is a good signed tree, then $\text{LIST-S-HOM}(\hat{H})$ is polynomial-time solvable.

We can explicitly state the dichotomy classification as follows.

Corollary 27. Let $\hat{H}$ be a signed tree. If any of the following conditions apply, then $\text{LIST-S-HOM}(\hat{H})$ is NP-complete.

1. $\hat{H}$ has no bicoloured loop, but there is a bicoloured (non-loop) edge and a unicoloured loop.
2. $\hat{H}$ has no bicoloured edge, but there is a red loop and a blue loop.
3. The bicoloured part $D_{\hat{H}}$ is not a bi-arc tree, i.e., contains a subgraph from Figures 3 or 4 of [11].
4. The blue part $B_{\hat{H}}$ is not a bi-arc tree, i.e., contains a subgraph from Figures 3 or 4 of [11].
5. $\hat{H}$ contains a signed tree from the family $T$.
6. The set of vertices of $\hat{H}$ with red (respectively blue, or at least blue, or bicoloured) loops induces a disconnected graph.

If none of the conditions apply, then $\text{LIST-S-HOM}(\hat{H})$ is polynomial-time solvable.

We also state the result in the more usual complementary way, where the polynomial cases are enumerated first. Note that here all the conditions are required to be satisfied to yield a polynomial case.

Corollary 28. Let $\hat{H}$ be a signed tree. If all of the following conditions apply, then $\text{LIST-S-HOM}(\hat{H})$ is polynomial-time solvable.

1. If $\hat{H}$ has no bicoloured edge, then all unicoloured loops are of the same colour.
2. If $\hat{H}$ has a bicoloured non-loop edge, then it has a bicoloured loop, or it has no loops at all.
3. The bicoloured part $D_{\hat{H}}$ is a bi-arc tree.
4. The blue part $B_{\hat{H}}$ is a bi-arc tree.
5. $\hat{H}$ contains no signed tree from the family $T$.
6. The vertices with red (respectively blue, respectively bicoloured) loops induce a connected subgraph of $\hat{H}$.

If at least one of the conditions fails, then $\text{LIST-S-HOM}(\hat{H})$ is NP-complete.

We now return to the proof of Theorem 26.

Proof. We show that for a good signed tree $\hat{H}$, the problem $\text{LIST-S-HOM}(\hat{H})$ is polynomial-time solvable. We may assume there is a bicoloured loop, else the result follows from Theorems 6 and 12.

Let $\hat{H}$ be a good signed tree obtained from a good reflexive caterpillar (with spine $v_1, v_2, \ldots, v_k$) by removing loops at a subset $S$ of leaves, and optionally replacing any bicoloured edges $v_i u$ by blue edges for the leaves $u \in S$. As in the case of reflexive trees,
we use a special min ordering of $\hat{H}$. This means that if a vertex $v_i$ (with $1 \leq i \leq d - 1$) has a non-loop neighbour $u$ connected by unicoloured edge, then $u$ is ordered to come after $v_{i+1}$ in the special min ordering. Now we can use our algorithm for reflexive trees, with the observation that if there is a negative cycle $C$ mapped to a unicoloured edge $v_j u$, then we can remove $u$ from lists of all vertices in $C$ and continue in modifying the images of such cycles.

Let $\hat{H}$ be a good signed 2-caterpillar (with spine $v_1, v_2, \ldots, v_k$), obtained from a bi-arc tree by replacing each edge and loop by bicoloured edge and loop (respectively), optionally adding unicoloured loops at leaves of $v_1$, and then adding blue edges from the spine or children of the spine to loopless leaves. We set $T = V(\hat{H})$ and $T' = V(D_2^\hat{H})$; moreover, we set $L = T \setminus T'$. It follows from Corollary 27 that all vertices of $L$ are loopless leaves in $T$ incident with exactly one blue edge. We also note that if two distinct vertices $a$ and $b$ in $T'$ are adjacent in $\hat{H}$, then they are adjacent by a bicoloured edge.

To prove our claim, we shall construct a suitable majority polymorphism. Recall that for a signed graph $\hat{H}$, the switching graph $S(\hat{H})$ is constructed as follows. We represent $\hat{H}$ as $(H, \pi)$ where the signature $\pi$ has all unicoloured non-loop edges blue (positive), and define $S(\hat{H})$ to be the edge-coloured graph $(H^+, \pi^+)$ in which each vertex $x$ of $H$ gives rise to two vertices $x, x'$ of $H^+$ and each edge $xy$ of $H$ gives rise to edges $xy$, $x'y'$ of the colour $\pi(xy)$ in $H^+$ and edges $x'y'$, $xy'$ of the opposite colour; this definition also applies to loops, by letting $x = y$. For any vertex $x$ of $H$, we shall denote by $x^+$ one of $x, x'$, and by $s(x^+)$ the other one of $x, x'$. A majority polymorphism of $(H^+, \pi^+)$ is a ternary mapping $F$ on the vertices of $H^+$ such that $F(x^+, y^+, z^+)$ is adjacent to $F(u^+, v^+, w^+)$ in blue (red) provided $x^+$ is adjacent to $u^+$ in blue (red), $y^+$ is adjacent to $v^+$ in blue (red), and $z^+$ is adjacent to $w^+$ in blue (red, respectively), and such that if two arguments from $x^+, y^+, z^+$ are equal, then the assigned value $F(x^+, y^+, z^+)$ is also equal to it. A semi-conservative majority polymorphism assigns $F(x^+, y^+, z^+)$ to be one of the values $x^+, y^+, z^+, s(x^+), s(y^+), s(z^+)$, and a conservative majority polymorphism assigns $F(x^+, y^+, z^+)$ to be one of the values $x^+, y^+, z^+$. As outlined in Section 3 if the edge-coloured graph $(H^+, \pi^+)$ admits a semi-conservative majority polymorphism, then the signed list homomorphism problem for $(H, \pi)$ is polynomial-time solvable. We shall in fact construct a conservative majority polymorphism of $(H^+, \pi^+)$. To construct a conservative majority polymorphism $F(x^+, y^+, z^+)$ for triples $(x^+, y^+, z^+)$ from $V(H^+)$, we will of course define values of triples with repetition to be the repeated value,

$$F(x^+, y^+, y^+) = F(y^+, y^+, x^+) = F(y^+, y^+, x^+) = y^+.$$ 

Now we partition the triples $(x^+, y^+, z^+)$ of vertices of $V(H^+)$ into two sets $R_1$ and $R_2$, where $R_1$ consists of those triples $(x^+, y^+, z^+)$ for which at most one of $x, y, z$ is in $L$, and $R_2$ consists of triples that have at least two of $x, y, z$ in $L$. (The vertices $x, y, z$ of $\hat{H}$ need not be distinct, as long as $x^+, y^+, z^+$ are distinct.) Note that two triples $(x^+_1, y^+_1, z^+_1), (x^+_2, y^+_2, z^+_2)$ that are coordinate-wise adjacent in $\hat{H}$ cannot both be in $R_2$, and if they are both in $R_1$, then there is a coordinate $t \in \{x, y, z\}$ such that $t_1 = t_2$, or the edge $t_1 t_2$ bicoloured in $\hat{H}$.

The definition of $F(x^+, y^+, z^+)$ will differ for triples with $(x, y, z) \in R_1$, where we explicitly describe the value $F(x^+, y^+, z^+)$, and for triples with $(x, y, z) \in R_2$, where we merely prove that a suitable value $F(x^+, y^+, z^+)$ exists.

First we consider the underlying unsigned tree of $\hat{H}$. It clearly contains all edges of $B_{\hat{H}}$, but it also contains loops that are red in $\hat{H}$. By Corollary 28 this tree (with vertex set $T$), which we also denote by $T$, is a bi-arc tree, and hence has a majority polymorphism $f$ [11].

We now describe the polymorphism $f$ from [11], assuming, as above, that the bi-arc tree $T$ is one of the trees in Figure 13.

In both cases, $T$ is a 2-caterpillar with spine $v_1, v_2, \ldots, v_k$, on which only $v_1$ has a loop,
and either there is only one additional loop on a child of $v_1$ (which may have children), or any number of loops on children of $v_1$ which must be leaves. (This is a slight change of notation from [11].) Each subtree $T_i$ rooted at a vertex $v_1$ of the spine is ordered by depth first search (in the case of $T_1$ giving higher priority to vertices with loops), and a total ordering of $T$ is obtained by concatenating these DFS orderings from $T_1$ to $T_2$ and so on. We also colour the vertices of $T$ by two colours, in a proper colouring ignoring the self-adjacencies due to the loops. Below we refer to vertices in $T_i$ other than the root $v_i$ as being inside $T_i$. The value $f(x, y, z)$ is defined as the majority of $x, y, z$ if two of the arguments $x, y, z$ are equal, and otherwise it is defined according to the following rules.

**Rule (A)** Assume $x, y, z$ are distinct and in the same colour class. Let $r(x), r(y), r(z)$ be the (not necessarily distinct) roots of the trees containing $x, y, z$ respectively, and let $v_m$ be the median of these vertices on the spine. Then $f(x, y, z)$ is the vertex from amongst $x, y, z$ in the tree $T_m$, and if there are more than one in $T_m$, it is the first vertex in the DFS ordering unless one of the following occurs, in which case it is the second vertex in the DFS ordering.

- All three vertices $x, y, z$ lie inside $T_m$ with $m \geq 2$;
- all three vertices $x, y, z$ lie inside $T_1$ and at most one of them has a loop;
- exactly two of $x, y, z$ lie inside $T_1$ and exactly one of them has a loop;
- exactly two of $x, y, z$ lie inside $T_1$, neither has a loop, exactly one of them is adjacent to the unique neighbour of $v_1$ with a loop, and the third vertex of $x, y, z$ is not $v_1$.

**Rule (B)** Assume $x, y, z$ are distinct but not all in the same colour class.

Then $f(x, y, z)$ is the first vertex in the DFS ordering of the two vertices in the same colour class, except when $\{x, y, z\}$ contains $v_1$ and at least one of its leaf neighbours with a loop, in which case $f(x, y, z) = v_1$.

We now use the above conservative majority $f$ on $T$ to define a conservative majority $F$ on triples in $R_1$. We say that a vertex $y$ dominates a vertex $x$ in $\hat{H}$ if any blue (or red) neighbour of $x$ is also blue (red respectively) neighbour of $y$.

**Rule (1)** Assume that at least two of $x^*, y^*, z^*$ are equal, say $y^* = z^*$. We define $F(x^*, y^*, y^*)$ to be the repeated value,

$$F(x^*, y^*, y^*) = F(y^*, x^*, y^*) = F(y^*, y^*, x^*) = y^*.$$

**Rule (2)** Assume that $x^*, y^*, z^*$ are distinct but two of $x, y, z$ are equal. Then for triples $(x^*, y^*, z^*)$ we define the value $F$ to be the first version of the repeated vertex, i.e.,

$$F(x^*, s(x^*), y^*) = F(x^*, y^*, s(x^*)) = F(y^*, x^*, s(x^*)) = x^*,$$

unless $x \in L$ or $x$ has a unicoloured loop in $\hat{H}$ and $y$ dominates $x$ in $\hat{H}$, in which case

$$F(x^*, s(x^*), y^*) = F(x^*, y^*, s(x^*)) = F(y^*, x^*, s(x^*)) = y^*.$$

(For example $F(x, x', y) = x'$ and $F(x', x, y) = x'$, but $F(x, x', y) = F(x', x, y) = y$ if $y$ dominates $x$ and $x$ has a unicoloured loop or is in $L$.)

**Rule (3)** Assume $x^*, y^*, z^*$ are distinct and also $x, y, z$ are distinct. For triples $(x^*, y^*, z^*)$, we define $F(x^*, y^*, z^*)$ to be the argument in the same coordinate as $f(x, y, z)$, except if $f(x, y, z) \in L$ and another vertex $t \in \{x, y, z\}$ dominates $f(x, y, z)$, in which case we define $F(x^*, y^*, z^*)$ to be the argument in the same coordinate as $t$. 

It is easy to check that if two triples \((x^*, y^*, z^*)\) and \((u^*, v^*, w^*)\) are coordinate-wise adjacent in blue (red) in \(S(\bar{H})\), then \((x, y, z)\) is adjacent to \((u, v, w)\) in \(T\) and hence \(f(x, y, z)\) is adjacent to \(f(u, v, w)\) in \(T\). We now check that we can also conclude that \(F(x^*, y^*, z^*)\) is adjacent to \(F(u^*, v^*, w^*)\) in blue (red respectively).

**Case 1.** \(x, y, z\) are distinct and \(u, v, w\) are distinct.

If \(f(x, y, z)\) and \(f(u, v, w)\) choose the same coordinate, then \(F(x^*, y^*, z^*)\) and \(F(u^*, v^*, w^*)\) also choose the same coordinate, and hence the values are adjacent in the right colour. (This remains true even if one or both of the choices \(F(x^*, y^*, z^*)\) and \(F(u^*, v^*, w^*)\) were modified by domination.) Otherwise, suppose without loss of generality that \(f(x, y, z) = x\) and \(f(u, v, w) = v\). Then the vertex \(x\) is adjacent in \(T\) to both \(u\) and \(v\) and hence is not a loop-free leaf, and similarly \(v\) is not a loop-free leaf. If \(x \neq v\), this means that the edge \(xv\) is bicoloured in \(\bar{H}\) and hence \(F(x^*, y^*, z^*) = x^*\) is adjacent to \(F(u^*, v^*, w^*) = v^*\) in both colours. If \(x = v\), the same argument applies if the loop \(xv\) is bicoloured, so let us assume it is unicoloured. In this situation, Proposition 24 implies that the vertex \(x = v\) must be a leaf child of \(v_1\) in \(T\), and \(u = y = v_1\). This is governed by the special case of Rule (B) in the definition of the majority polymorphism \(f\), which implies that we would have \(f(x, y, z) = v_1\), contradicting \(f(x, y, z) = x\), so this case does not occur.

**Case 2.** \(u, v, w\) are distinct but \(x, y, z\) are not distinct, say \(x = y\) (but perhaps \(x^* \neq y^*\)).

This means that \(f(u, v, w)\) is adjacent to \(x = f(x, y, z)\) in \(T\), and \(x\) is not a loop-free leaf since it is adjacent to both \(u\) and \(v\). We now observe that if \(F(u^*, v^*, w^*)\) was chosen in the same coordinate as \(f(u, v, w)\), then \(x\) is in \(T'\), and otherwise it was chosen in the same coordinate as some \(t \in T'\) (we used Rule (3)). Hence \(f(u, v, w)\) or \(t\) is adjacent to \(x\) by a bicoloured edge in \(\bar{H}\), and \(F(u^*, v^*, w^*)\) adjacent to \(x^*\) and to \(s(x^*)\) in both colours. (Note that \(F(x^*, y^*, z^*) = x^*\) regardless of whether \(y^* = x^*\) or \(y^* = s(x^*)\).)

**Case 3.** Each triple \(x, y, z\) and \(u, v, w\) has exactly one repetition.

Suppose first that the repetition is in different positions, say \(x = y\) and \(v = w\). Then \(f(x, y, z) = x\) is adjacent to \(f(u, v, w) = v\) in \(T\). If \(x \neq v\), then the edge \(xv\) is bicoloured in \(\bar{H}\), and \(F(x^*, y^*, z^*) = x^*\) or \(F(x^*, y^*, z^*) = s(x^*)\) and \(F(u^*, v^*, w^*) = v^*\) or \(F(u^*, v^*, w^*) = s(v^*)\) are adjacent in both colours. If \(x = v\), then the same argument applies if the loop is bicoloured, and if it is unicoloured, then Proposition 24 implies that \(u = z\) and \(u\) has a bicoloured loop and dominates \(x\), whence \(F(x^*, y^*, z^*) = x^*\) and \(F(u^*, v^*, w^*) = u^*\) and the adjacency is correct. On the other hand, if the repetition is in the same positions, say \(x = y\), then \(F(x^*, y^*, z^*) = x^*\) is adjacent to \(F(u^*, v^*, w^*) = u^*\) by the definition of \(F\), and hence the edge has the correct colour.

**Case 4.** One triple has all vertices the same, say, \(x = y = z\) (but possibly \(x^* \neq y^*\)).

If we also have \(u = v = w\), then by the pigeon principle some coordinate contains both \(F(x^*, y^*, z^*)\) in \((x^*, y^*, z^*)\) and \(F(u^*, v^*, w^*)\) in \((u^*, v^*, w^*)\), and so we have the correct adjacency. If \(x\) is joined to \(u\) by a bicoloured edge in \(\bar{H}\), then \(F(x^*, y^*, z^*)\) is joined to \(F(u^*, v^*, w^*)\) with the correct adjacency (even in the domination case of Rule 3). As both \(x, u \not\in L\), the only way for the edge joining them to be unicoloured, is \(x = u\) and the edge is a unicoloured loop. In this case by Proposition 24 \(w\) dominates \(u\) and is joined to \(u = x\) with a bicoloured edge in \(\bar{H}\), again ensuring the right adjacency.

Now we prove that one can extend the definition of \(F\) to \(R_2\) so that it remains a polymorphism. (It is of course possible to define each \(F(u^*, v^*, w^*)\) for \((u^*, v^*, w^*) \in R_2\) directly, but we found the arguments become more transparent if we only verify that a suitable choice for \(F(u^*, v^*, w^*)\) is always possible.)

Consider first values \(F(x^*, y^*, z^*)\) with all three vertices \(x, y, z\) in \(L\). This means that \(x\) is incident in \(\bar{H}\) with only one (necessarily blue) edge, say \(xx_1\), and similarly for blue edges for \(y, z\).
Thus in the switching graph \( S(\tilde{H}) \) the vertex \( x \) is incident with only one blue edge, namely \( xx_1 \), and one red edge, namely \( xx_1' \), and similarly for \( y \) and for \( y', z, z' \). Note that \( (x_1, y_1, z_1) \in R_1 \) because two vertices of \( L \) are never adjacent. To choose the value of \( F(x_1^*, y_1^*, z_1^*) \), we only need to take into account the existing values of \( F(x_1^{**}, y_1^{**}, z_1^{**}) \), where \( x_1^{**} \) is also \( x_1 \) or \( x_1' \), and similarly for \( y_1^{**}, z_1^{**} \). For example, \( (x, y', z') \) is coordinate-wise adjacent in blue only to \( (x, y_1', z_1') \) and in red only to \( (x_1', y_1, z_1) \), and the choices of \( F(x_1, y_1', z_1') \) and \( F(x_1', y_1, z_1) \) occur in the same coordinate, by the definition of \( F \) on \( R_1 \); if, say, \( F(x_1, y_1', z_1') = x_1 \) and \( F(x_1', y_1, z_1) = x_1' \), then setting \( F(x, y', z') = x \) ensures that \( F(x,y',z') \) is adjacent to \( F(x_1,y_1',z_1') = x_1 \) in blue and to \( F(x_1', y_1, z_1) = x_1' \) in red, as required. Thus in general we can choose the value \( F(x^*, y^*, z^*) \) in the same coordinate as \( F(x_1^{**}, y_1^{**}, z_1^{**}) \), and satisfy the polymorphism property. (Note that this argument applies even if the vertices \( x, y, z \) are not distinct.)

It remains to consider the case when exactly two of \( x, y, z \) belong to \( L \), say \( x \in L \) and \( y \in L \), with unique (blue) neighbours \( x_1 \) and \( y_1 \) in \( \tilde{H} \), and \( z \notin L \), with neighbours \( z_1, \ldots, z_p \). We want to show that there is a suitable value for each \( F(x^*, y^*, z^*) \) that maintains the polymorphism property. In the proofs below, we use the fact that \( (x^*, y^*, z^*) \) is coordinate-wise adjacent in at least blue to each \( (x_1^{**}, y_1^{**}, z_1^{**}) \) and possibly also \( (x_1^{**}, y_1^{**}, z_1^{**}) \) (if the edge \( zz_i \) is bicoloured), and adjacent in at least red to each \( (s(x_1^{**}), s(y_1^{**}), s(z_1^{**})) \) and possibly also \( (s(x_1^{**}), s(y_1^{**}), s(z_1^{**})) \) (if the edge \( zz_i \) is bicoloured). In any event, we again denote the relevant triples by \( (x_1^{**}, y_1^{**}, z_1^{**}) \).

Suppose that \( x, y, z \) are of the same colour. We observe that \( x \) and \( y \) cannot lie on the spine, since they are in \( L \).

Consider first the case that \( x \) and \( y \) lie inside the same tree \( T_r \). Recall that we say 'inside' to mean \( x \) and \( y \) are not on the spine; thus vertices \( x_1 \) and \( y_1 \) also belong to \( T_r \), and so \( T_r \) is the median tree. If \( z \) also lies inside \( T_r \), the argument is similar to previous cases where \( F(x^*, y^*, z^*) \) is chosen according to the unique neighbours of \( x^*, y^*, z^* \). If no neighbour \( z_i \) of \( z \) is in \( T_r \), then each value \( F(x_1^{**}, y_1^{**}, z_1^{**}) \) is either \( x_1^{**} \) or \( y_1^{**} \) independently of the location of \( z_i \), and hence choosing correspondingly \( F(x^*, y^*, z^*) = x^* \) or \( y^* \) will ensure the polymorphism property. If some neighbour \( z_i \) of \( z \) lies in \( T_r \), \( r > 1 \), then \( z \) is the root of \( T_{r-1} \), or of \( T_r \), or of \( T_{r+1} \), and in this case, we can choose \( F(x^*, y^*, z^*) = z^* \). Indeed, in this case, \( zz_i \) is bicoloured, and \( z_i = x_1 = y_1 \) (if \( z \) is in \( T_{r-1} \) or \( T_{r+1} \)) or \( x_1, y_1 \) are also bicoloured (if \( z \) is in \( T_r \)). If \( r = 1 \), then in addition to the previous case the vertices \( z, z_1, \ldots, z_p \) can have loops (the vertices \( x, y, x_1, y_1 \) do not have loops). Since \( x_1 \) and \( y_1 \) have the same colour, if the colour of \( z_i \) is different (when \( z = z_i \), we have the value of \( F(x_1^{**}, y_1^{**}, z_1^{**}) \) equal to the first or second coordinate, and we can choose \( F(x^*, y^*, z^*) \) accordingly. Note that these arguments apply also when \( y^* = s(x^*) \), i.e., \( x = y \).

If \( x \in T_r \), \( y \in T_s \) with \( r \neq s \), the arguments are similar. If no neighbour \( z_j \) of \( z \) lies in \( T_r \) or \( T_s \), then we can choose \( F(x^*, y^*, z^*) = x^* \) or \( y^* \) or \( z^* \), depending on which is the median tree, and if some \( z_j \) lies in, say, \( T_r \), then we can choose \( F(x^*, y^*, z^*) = x^* \) or \( z^* \) as above.

Next we consider the case when \( x, y, z \) do not have the same colour. We may assume that \( x, y \) have different colours, since if \( x, y \) have the same colour and \( z \) has a different colour then any relevant \( F(x_1^{**}, y_1^{**}, z_1^{**}) \) is \( x_1^{**} \) or \( y_1^{**} \) regardless of \( z_i \) and we can choose \( F(x^*, y^*, z^*) \) accordingly. However, if \( z \) has a loop, then we need to consider also \( F(x_1^{**}, y_1^{**}, z_1^{**}) \), which could be \( z^{**} \) or \( s(z^{**}) \) if \( z \) is the vertex \( v_1 \); in that case we can set \( F(x^*, y^*, z^*) = z^* \).

Thus assume without loss of generality that \( x, z \) have the same colour, but the colour of \( y \) is different. Then unless \( z \) has a loop, \( F(x_1^{**}, y_1^{**}, z_1^{**}) \) is \( x_1^{**} \) or \( z_1^{**} \), depending on whether \( z_i \) precedes or follows \( x_1 \) in the DFS ordering. If all neighbours \( z_i \) precede \( x_1 \), or all follow
x_1$, the uniform choice of $F(x_1^+, y_1^+, z_1^+)$ allows one to choose $F(x^*, y^*, z^*)$ accordingly. The only situation when some $z_j$ precedes $x_1$ and another $z_j$ follows $x_1$ occurs when $z$ is the root of the tree $T_\ell$ containing $x$. It is easy to see that in that case we can set $F(x^*, y^*, z^*) = z^*$. Finally, when $z$ has a loop, then we also need to consider $F(x_1^+, y_1^+, z_1^+) = z^+$ and we set $F(x^*, y^*, z^*) = z^*$.

8 Irreflexive signed graphs

Irreflexive signed graphs are in some sense the core of the problem. By Theorem 6, the problem List-S-Hom($\hat{H}$) is NP-complete unless the underlying graph $H$ is bipartite. There is a natural transformation of each general problem to a problem for a bipartite irreflexive signed graph, akin to what is done for unsigned graphs in [11]; this is nicely explained in [21].

However, for bipartite $H$, we don’t have a combinatorial classification beyond the case of trees $H$, except in the case $\hat{H}$ has no bicoloured edges or loops (when Theorem 6 applies), or when $\hat{H}$ has no unicoloured edges or loops (when the problem essentially concerns unsigned graphs and thus is solved by [11]). Therefore we may assume that both bicoloured and unicoloured edges or loops are present; we focus in this section on those bipartite irreflexive signed graphs $\hat{H}$ in which the unicoloured edges form simple structures, such as paths and cycles. We start with irreflexive signed graphs in which the unicoloured edges form a spanning path.

We say that an irreflexive signed graph $\hat{H}$ is path-separable if the unicoloured edges of $\hat{H}$ form a Hamiltonian path $P$ in the underlying graph $H$. We may assume the edges of $P$ are all blue. In other words, all the edges of the Hamiltonian path $P$ are blue, and all the other edges of $\hat{H}$ are bicoloured. Recall that the distinction between unicoloured and bicoloured edges is independent of switching, thus such a Hamiltonian path $P = v_1v_2\ldots v_n$ is unique, if it exists.

We first observe that for any irreflexive signed graph $\hat{H}$, the problem List-S-Hom($\hat{H}$) is NP-complete if the underlying graph $H$ contains a cycle, since then the s-core of $\hat{H}$ has at least three edges. Moreover, we now show that List-S-Hom($\hat{H}$) is also NP-complete if $H$ contains an induced cycle of length greater than four. Indeed, it suffices to prove this if $H$ is an even cycle of length $k > 4$. If all edges of $H$ are unicoloured, then the problem is NP-complete by Theorem 6 since an irreflexive cycle of length $k > 4$ is not a bi-arc graph. If all edges of the cycle $H$ are bicoloured, then we can easily reduce from the previous case. If $H$ contains both unicoloured and bicoloured edges, then $\hat{H}$ contains an induced subgraph of type a) or b) in the family $F$ in Figure 3 and List-S-Hom($\hat{H}$) is NP-complete by Lemma 10. (There are cases when the subgraphs are not induced, but the chains from the proof of Lemma 10 are still applicable.)

We further identify two cases of NP-complete List-S-Hom($\hat{H}$). An alternating 4-cycle is a 4-cycle $v_1v_2v_3v_4$ in which the edges $v_1v_2$, $v_3v_4$ are bicoloured and the edges $v_2v_3$, $v_4v_1$ unicoloured. An 4-cycle pair consists of 4-cycles $v_1v_2v_3v_4$ and $v_5v_6v_7v_8$, sharing the vertex $v_1$, in which the edges $v_1v_2$, $v_1v_5$ are bicoloured, and all other edges are unicoloured. An alternating 4-cycle has the chain $U = v_1, v_4, v_3; D = v_1, v_2, v_3$, and a 4-cycle pair has the chain $U = v_1, v_4, v_3, v_2, v_1; D = v_1, v_5, v_6, v_7, v_1$. Therefore, if a signed graph $\hat{H}$ contains an alternating 4-cycle or a 4-cycle pair as an induced subgraph, then the problem List-S-Hom($\hat{H}$) is NP-complete. Note that the latter chain requires only $v_2v_6$ and $v_3v_5$ to be non-edges. The problem remains NP-complete as long as these edges are absent; all other edges with endpoints in different 4-cycles can be present. If both $v_2v_6$ and $v_3v_5$ are bicoloured edges, then there is an alternating 4-cycle $v_2v_3v_5v_6$. Thus we conclude that the problem is NP-complete.
if $\hat{H}$ contains a 4-cycle pair as a subgraph (not necessarily induced), unless exactly one of $v_1v_3$ or $v_2v_6$ is a bicoloured edge.

From now on we will assume that $\hat{H}$ is a path-separable signed graph with the unicoloured edges (all blue) forming a Hamiltonian path $P = v_1, \ldots, v_n$. We will assume further that LIST-S-Hom($\hat{H}$) is not NP-complete, and derive information on the structure of $\hat{H}$. In particular, the underlying graph $H$ is bipartite and does not contain any induced cycles of length greater than 4, and $\hat{H}$ does not contain an alternating 4-cycle or a 4-cycle pair; more generally, $\hat{H}$ does not contain a chain. If $\hat{H}$ has no bicoloured edges (and hence no edges not on $P$), then LIST-S-Hom($\hat{H}$) is polynomially solvable by Theorem 6 since a path is a bi-arc graph. If there is a bicoloured edge in $\hat{H}$, then we may assume there is an edge $v_iv_{i+3}$, otherwise there is an induced cycle of length greater than 4.

A block in a path-separable signed graph $\hat{H}$ is a subpath $v_iv_{i+1}v_{i+2}v_{i+3}$ of $P$, with the bicoloured edge $v_iv_{i+3}$. The previous paragraph concluded that $\hat{H}$ must contain a block. Note that if $v_iv_{i+1}v_{i+2}v_{i+3}$ is a block, then $v_{i+1}v_{i+2}v_{i+3}$ cannot be a block: in fact, $v_{i+1}v_{i+4}$ cannot be a bicoloured edge, otherwise $\hat{H}$ would contain an alternating 4-cycle. However, $v_{i+2}v_{i+3}v_{i+4}v_{i+5}$ can again be a block, and so can $v_{i+4}v_{i+5}v_{i+6}v_{i+7}$, etc. If both $v_iv_{i+1}v_{i+2}v_{i+3}$ and $v_{i+2}v_{i+3}v_{i+4}v_{i+5}$ are blocks then $v_{i+5}$ must be a bicoloured edge, otherwise $v_iv_{i+3}v_{i+2}v_{i+5}$ would induce a signed graph of type a) in family $F$ from Figure 3. A segment in $\hat{H}$ is a maximal subpath $v_iv_{i+1} \ldots v_{i+2j+1}$ of $P$ with $j \geq 1$ that has all bicoloured edges $v_{i+e}v_{i+e+3}$, where $e$ is even, $0 \leq e \leq 2j-2$. (A maximal subpath is not properly contained in another such subpath.) Thus each subpath $v_{i+e}v_{i+e+1}v_{i+e+2}v_{i+e+3}$ of the segment is a block, and the segment is a consecutively intersecting sequence of blocks; note that it can consist of just one block. Two segments can touch as the second and third segment in Figure 3, or leave a gap as the first and second segment in the same figure.

In a segment $v_iv_{i+1} \ldots v_{i+2j+1}$ we call each vertex $v_{i+e}$ with $0 \leq e \leq 2j-2$ a forward source, and each vertex $v_{i+o}$ with $3 \leq o \leq 2j+1$ a backward source. Thus forward sources are the beginning vertices of blocks in the segment, and the backward sources are the ends of blocks in the segment. If $a < b$, we say the edge $v_av_b$ is a forward edge from $v_a$ and a backward edge from $v_b$. In this terminology, each forward source has a forward edge to its corresponding backward source. Because of the absence of a signed graph of type a) in family $F$ from Figure 3, we can in fact conclude, by the same argument as in the previous paragraph, that each forward source in a segment has forward edges to all backward sources in the segment.

We say that a segment $v_iv_{i+1} \ldots v_{i+2j+1}$ is right-leaning if $v_iv_{i+e}v_{i+e+1}$ is a bicoloured edge for all $e$ is even, $0 \leq e \leq 2j-2$, and all odd $o \geq 3$; and we say it is left-leaning if $v_{i+2j+1-e}v_{i+2j+1-e-o}$ is a bicoloured edge for all $e$ even, $0 \leq e \leq 2j-2$ and all odd $o \geq 3$. Thus a in a right-leaning segment each forward source has all possible forward edges (that is, all edges to vertices of opposite colour in the bipartition, including vertices with subscripts greater than $i+2j+1$). The concepts of left-leaning segments, backward sources and backward edges are defined similarly.

We say that a path-separable signed graph $\hat{H}$ is right-segmented if all segments are right-leaning, and there are no edges other than those mandated by this fact. In other words, each forward edge has all possible forward edges, and each vertex which is not a forward source has no forward edges. Similarly, we say that a path-separable signed graph $\hat{H}$ is left-segmented if all segments are left-leaning, and there are no edges other than those mandated by this fact. In other words, each backward edge has all possible backward edges, and each vertex which is not a backward source has no backward edges. Finally, $\hat{H}$ is left-right-segmented if there is a unique segment $v_iv_{i+1} \ldots v_{i+2j+1}$ that is both left-leaning
and right-leaning, all segments preceding it are left-leaning, all segments following it are right-leaning, and moreover there are additional bicoloured edges \( v_{i-e}v_{i+2j+o} \) for all even \( e \geq 2 \) and all odd \( o \geq 3 \), but no other edges. In other words, vertices \( v_1, v_2, \ldots, v_{i+2j+1} \) induce a left-segmented graph, vertices \( v_i, v_{i+1}, \ldots, v_n \) induce a right-segmented graph, and in addition to the edges this requires there are all the edges joining \( v_{i-e} \) from \( v_1, \ldots, v_{i-1} \) to \( v_{i+o} \) from \( v_{i+2j+2}, \ldots, v_n \), with even \( e \) and odd \( o \). A segmented graph is a path-separable signed graph that is right-segmented or left-segmented or left-right-segmented.

See Figure 16 there are three segments, the left-leaning segment \( v_5v_{10}v_7v_9v_{15} \), the left- and right-leaning segment \( v_{12}v_{13}v_{14}v_{15} \), and the right-leaning segment \( v_{15}v_{16}v_{17}v_{18}v_{19}v_{20} \). Thus this is a left-right-segmented signed graph.

**Theorem 29.** Let \( \hat{H} \) be a path-separable signed graph. Then \( \text{List-S-Hom}(\hat{H}) \) is polynomial-time solvable if \( H \) is switching equivalent to a segmented signed graph \( \hat{H} \). Otherwise, the problem is \( \text{NP-complete} \).

**Proof.** Assume as above that \( \hat{H} \) is a path-separable signed graph for which \( \text{List-S-Hom}(\hat{H}) \) is not \( \text{NP-complete} \), with the unicoloured edges all blue and forming the Hamiltonian path \( P = v_1, \ldots, v_n \). As noted above, there are bicoloured edges forming segments, and possibly some other bicoloured edges.

Consider two consecutive segments, a segment \( S \), ending with the block \( v_i, v_{i+1}, v_{i+2}, v_{i+3} \), and a segment \( S' \) beginning with the block \( v_j, v_{j+1}, v_{j+2}, v_{j+3} \), where \( i + 3 \leq j \). Note that there can be no bicoloured edge joining two vertices from the set \( \{v_{i+1}, \ldots, v_{j+2}\} \), because the segments \( S, S' \) are maximal and consecutive. (If there is any edge joining two vertices of that set, there would have to be one forming a 4-cycle with the unicoloured edges, since the underlying graph has no induced cycles longer than 4.) We emphasize that this crucial observation will be repeatedly used in the arguments in the next three paragraphs, usually without specifically mentioning it.

We claim that either \( v_i \) (the last forward source of the segment \( S \)) has forward edges to all \( v_{i+3}, v_{i+5}, v_{i+7}, \ldots, v_s \) for some \( s > j + 1 \), or symmetrically, \( v_{j+3} \) (the first backward source of the next segment \( S' \)) has backward edges to all \( v_{j-2}, v_{j-4}, \ldots, v_t \) for some \( t < i + 2 \). In the former case we say that \( S \) precedes \( S' \), in the latter case we say that \( S' \) precedes \( S \).

Suppose first that \( i \) and \( j \) have the same parity (are both even or both odd). There must be other bicoloured edges, otherwise there is a signed graph of type a) from the family \( \mathcal{F} \) in Figure 3 induced on the vertices \( v_i, v_{i+3}, v_{i+4}, \ldots, v_j, v_{j+3} \), and hence a chain in \( \hat{H} \). Therefore, using the above crucial observation, there must be extra edges incident with \( v_i \) or \( v_{j+3} \). Moreover, as long as there is no edge \( v_i v_{j+3} \), there would always be an induced subgraph of type a) from the family \( \mathcal{F} \). On the other hand, if \( v_i v_{j+3} \) is an edge, then there is a chain with \( U = v_i, v_{i+1}, v_{i+2}, v_{i+3}, v_j, v_{j+3} \) and \( D = v_i, v_{j+3}, v_{j+2}, v_{j+1}, v_{j+2}, v_{j+3} \), unless \( v_i v_{j+3} \) or \( v_i v_{j+1} \) is an edge. (There is no edge \( v_{i+3}v_{j+2} \) by our crucial observation above.) Note that both \( v_{i+2}v_{j+3} \) and \( v_i v_{j+1} \) cannot be edges, because of the chain \( U = v_i, v_{i+1}, v_{i+2}, v_{j+3} \), \( D = \)
list homomorphism problems for signed graphs

Assume that $v_i v_{j+1}$ is an edge. Now we can repeat the argument: there would be a chain with $U = v_i, v_{i+1}, v_{i+2}, v_{i+3}, v_i, v_{j+1}$ and $D = v_i, v_{j+1}, v_{j+2}, v_{j+3}, v_i, v_{j+1}$, unless $v_i v_{j+1}$ is an edge. (In this case, we don’t need to consider $v_{i+2} v_{j+1}$, since both lie in $v_i, v_{i+1}, v_{j+2}$.) We can continue this way until this argument implies the already existing edge $v_i v_{j+3}$, and conclude that $v_i$ is adjacent to all $v_{j+3}, v_{j+1}, v_{j+1}, \ldots, v_{i+3}$, which proves the claim with $s = j + 3$. If $v_{i+2} v_{j+3}$ is an edge, we conclude symmetrically that the claim holds for $t = i$.

Now assume that the parity of $i$ and $j$ is different. This happens, for instance, when $i + 3 = j$: in this case, we would have a 4-cycle pair unless one of $v_i v_{j+2}, v_{i+1} v_{j+3}$ is an edge. Both cannot be edges, as there would be an alternating 4-cycle. This verifies the claim when $i + 3 = j$. Otherwise, there again is a signed graph of type a) from the family $F$ in Figure 3, induced on the vertices $v_i, v_{i+3}, v_{i+4}, \ldots, v_j, v_{j+3}$, so there must be extra edges incident with $v_i$ or $v_{j+3}$. Moreover, there would always remain such an induced subgraph unless there is a vertex $v_p$ with $i + 3 \leq p \leq j$ that is adjacent to both $v_i$ and $v_{j+3}$. Thus let $v_p$ be such a vertex.

We first show that $p$ can be chosen to be $j$ or $i + 3$, i.e., that $v_j$ is adjacent to $v_i$, or $v_{j+3}$ is adjacent to $v_{i+3}$. Indeed, if $v_i v_j, v_{i+3} v_{j+3}$ are not edges, then $v_i v_{j+2}$ must also not be an edge (else we obtain a cycle of length greater than 4), and we have the chain $U = v_i, v_{i+1}, v_{i+2}, v_{i+3}, v_i, v_p, v_{j+3}, D = v_i, v_p, v_{j+3}, v_{j+2}, v_{j+1}, v_{j+2}, v_{j+3}$. (Recall that we are still using the crucial observation that there is no bicoloured edge joining two vertices from the set $v_i, v_{i+1}, \ldots, v_{j+2}$.)

Consider now the case that $p = j$ (or symmetrically $p = i + 3$). Since $v_i v_p$ is an edge and there are no induced cycles of length greater than 4, the vertex $v_i$ must also be adjacent to $v_{j-2}, v_{j-4}, \ldots, v_{i+5}$. Moreover, using again the crucial observation, $v_j$ is also adjacent to $v_{j+2}$, as otherwise we would have the chain $U = v_i, v_{i+1}, v_{i+2}, v_{i+3}, v_{i+4}, v_j, v_j$, and $D = v_i, v_{j+1}, v_{j+2}, v_{j+1}, v_j$. Thus the claim holds, with $s = j + 2$. In the case $p = i + 3$, we obtain a symmetric situation, proving the claim with $t = i + 1$.

We conclude that for any two consecutive segments, exactly one precedes the other. For technical reasons, we also introduce two auxiliary segments, calling the other segments normal. If the first normal segment $S$ of $\tilde{H}$ starts at $v_i$ with $i > 2$, we introduce the left end-segment to consist of the vertices $v_1, v_2, \ldots, v_i$. We say that the left end-segment precedes $S$ if there is no edge $v_2 v_{i+3}$, and we say that $S$ precedes the left end-segment if $v_2 v_{i+3}$ is an edge. Similarly, if the last normal segment $S'$ ends at $v_k$ with $k < n - 1$, the right end-segment consists of the vertices $v_k, v_{k+1}, \ldots, v_n$. The right end-segment precedes $S'$ if $v_k v_{k+3} v_{k+2}$ is not an edge, and $S'$ precedes the right end-segment if $v_k v_{k+3} v_{k+2}$ is an edge. Then it is still true that for any two consecutive segments, one precedes the other.

Suppose that we have the special situation where each segment (including the end-segments) precedes the next segment. Consider again the last normal segment $S'$, ending in block $v_{k-3}, v_{k-2}, v_{k-1}, v_k$. We first note that since there are no induced cycles of length greater than 4, and no blocks after the block $v_{k-3}, v_{k-2}, v_{k-1}, v_k$, there cannot be any forward edges from $v_{k-1}, v_k, v_{k+1}, \ldots$. By the same argument and the absence of alternating 4-cycles, there are no forward edges from $v_{k-2}$ either.

Our special assumption implies that $v_{k-3} v_{k+2}$ is an edge. Then $v_{k-3}$ has also an edge to $v_{k+4}$, otherwise we have a signed graph of type b) from family $F$ in Figure 3, induced on the vertices $v_{k-1}, v_{k-2}, v_{k-3}, v_{k+2}, v_{k+3}, v_{k+4}$. It is easy to check that the subgraph is induced because otherwise there would be either another block, or an alternating 4-cycle. Then we argue similarly that $v_{k-3}$ has also an edge to $v_{k+6}$, and so on, concluding by induction that the last forward source $v_{k-3}$ has all possible forward edges, and that no vertex after $v_{k-3}$
The proof is analogous to the preceding paragraph. Consider for instance a segment even subscripts signed graph. We give a min ordering of the vertices. Consider two vertices graph that is not segmented. We must be present. This completes the proof of NP-completeness for any path-separable signed graph.

After we have the edges signed graph is both left-leaning and right-leaning. Then we must have the edge e preceding the next block, its last forward source, v has all possible forward edges. Thus the last segment S is right-leaning, and there are no other forward edges (starting in its vertices or later) than those mandated by this fact.

We proceed by induction from the last segment to the first segment to show that in this special situation all segments are right-leaning and there are no other forward edges at all. The proof is analogous to the preceding paragraph. Consider for instance a segment S ending in block v_i, v_{i+1}, v_{i+2}, v_{i+3}; since it precedes the next block, its last forward source, v_i has all possible forward edges until v_s where s > j + 1 and the next segment begins with v_{i+1}. Now the arguments can be repeated, starting avoiding a signed graph of type b) from family F in Figure 3 induced on the vertices v_{i+2}, v_{i+1}, v_i, v_s, v_{s+1}, v_{s+2}. Finally, the vertices in the left end-segment cannot have any forward edges, as the absence of other blocks and of induced cycles of length greater than 4 implies there would have to be an edge vF−1vF where vF is the first backward source, contrary to the assumption that the left end-segment precedes the first normal segment.

Thus we have proved that if each segment precedes the next segment, then H is a right-segmented graph. By symmetric arguments, we obtain the case of left-segmented signed graphs by assuming that each segment precedes the previous segment. It remains to consider the cases where some segment precedes, or is preceded by, both its left and right neighbours.

It turns out that the case where two segments S_1 and S_3 both precede the intermediate segment S_2 is impossible. Suppose S_2 has vertices v_a, v_{a+1}, ..., v_b; in particular, this implies that v_av_b is an edge. Since each segment before S_2 precedes the next segment, the previous arguments apply to the portion of the vertices before S_2, and in particular the vertex v_{a-2} is not a forward source, hence has no forward edges; therefore v_{a-2} is not adjacent to v_b. By a symmetric argument, there is no edge v_av_{b+2}. There is also no edge v_{a-1}v_{b+1} because it would form an alternating cycle with v_av_b. Therefore, v_{a-2}, v_{a-1}, v_a, v_b, v_{b+1}, v_{b+2} induce a signed graph of type b) from family F in Figure 3, a contradiction.

We conclude that if S_1 precedes the next segment S_2, then S_2 must precede the following segment S_3, and so on, and similarly if S_1 precedes the previous segment S_2.

Hence it remains only to consider the situation where a unique segment S_2, with vertices v_a, v_{a+1}, ..., v_b precedes both its left neighbour S_1 and its right neighbour S_3 and to the left of S_1 each segment precedes its left neighbour, and to the right of S_3 each segment precedes its right neighbour. This implies that all segments before S_2 are left-leaning, all segments after S_2 are right-leaning, while S_2 is both left-leaning and right-leaning. To prove that in this situation the signed graph H is left-right-segmented, we show that all edges v_{a−2}v_{a+o} are present, with e even and o odd. This is obvious when e = 0 and o ≤ b − a, by the observations following the definition of a segment. We also have the edges v_{a−2}v_{b+2}, v_{a−2}v_b since the segment S_2 is both left-leaning and right-leaning. Then we must have the edge v_{a−2}v_{b+2} else there would be the chain U = v_a, v_{a−1}, v_{a−2}, v_b, D = v_a, v_{b+2}, v_{b+1}, v_b. We have the edge v_{a−4}v_{b+4} since v_{a−4} is a forward source, and thus we must also have the edge v_{a−2}v_{b+4}, otherwise there is the chain U = v_a, v_{a−1}, v_{a−2}, v_{b+2}, D = v_a, v_{b+4}, v_{b+3}, v_{b+2}. Continuing this way by induction on e + o we conclude that all edges v_{a−e}v_{a+o}, with e even and o odd, must be present. This completes the proof of NP-completeness for any path-separable signed graph that is not segmented.

We now describe polynomial-time algorithms for LIST-S-HOM(H) when H is a segmented signed graph. We give a min ordering of the vertices. Consider two vertices v_x and v_y with even subscripts x < y, and each with a forward bicoloured edge. Then all forward neighbours
of \(v_g\) are also forward neighbours of \(v_x\), and all backward neighbours of \(v_x\) are also backward neighbours of \(v_g\). Vertices \(v_z\) with no forward bicoloured edges have backward edges to all vertices with forward bicoloured edges from vertices \(v_t\) with \(t\) odd and \(t < z\). We now order the vertices with even subscripts as follows: first we take vertices with forward bicoloured edges in increasing order of subscripts, then we take the remaining vertices in the decreasing order of subscripts. The same ordering is applied on the vertices with odd subscripts. It now follows from our observations that this is a min ordering, and for every vertex the bicoloured edges come before the unicoloured edges. Thus the conclusion holds by Lemma 14. ▶

As an application of Theorem 29, we now consider irreflexive signed graphs in which the unicoloured edges form a spanning cycle. We say that an irreflexive signed graph \(\hat{H}\) is cycle-separable if the unicoloured edges of \(\hat{H}\) form a Hamiltonian cycle \(C\) in the underlying graph \(H\). We may no longer assume the edges of \(C\) are all blue, see below. In other words, we have a Hamiltonian cycle \(C\) whose edges are all unicoloured, and all the other edges of \(\hat{H}\) are bicoloured. (The Hamiltonian cycle \(P = v_1v_2\ldots v_n\) is still unique, if it exists.)

We first introduce three cycle-separable signed graphs for which the list homomorphism problem will turn out to be polynomial-time solvable. The signed graph \(\hat{H}_0\) is the 4-cycle with all edges unicoloured blue. The signed graph \(\hat{H}_1\) consists of a a blue path \(t, s_1, s_2, w\), a red path \(t = t_0, t_1, t_2, t_3 = w\), together with a bicoloured edge \(tw\). The signed graph \(\hat{H}_\ell\) consists of a blue path \(t, s_1, s_2, w\), a blue path \(t = t_0, t_1, t_2, \ldots, t_s = w\) (with \(s\) odd), and all bicoloured edges \(t_it_j\) with even \(i\) and odd \(j > i + 1\). (Note that this includes the edge \(tw\).) These three cycle separable signed graphs are illustrated in Figure 17. Note that if the subscript \(s\) is greater than \(o\) then it is odd, and that \(H_1\) and \(H_3\) both have 6 vertices and differ only in the colours of the Hamiltonian cycle \(C\) of unicoloured edges: \(H_1\) has the cycle \(C\) unbalanced, and \(H_3\) has the cycle \(C\) balanced.

Figure 17 The cycle-separable signed graphs \(\hat{H}_0, \hat{H}_1,\) and \(\hat{H}_\ell\) with \(\ell \geq 3\) odd.

**Theorem 30.** Let \(\hat{H}\) be a cycle-separable signed graph. Then the problem \(\text{List-S-Hom}(\hat{H})\) is polynomial-time solvable if \(\hat{H}\) is switching equivalent to \(\hat{H}_0\), or to \(\hat{H}_1\), or to \(\hat{H}_\ell\) for some odd \(\ell \geq 3\). Otherwise, the problem is NP-complete.

**Proof.** Suppose \(\hat{H}\) is a cycle-separable signed graph for which \(\text{List-S-Hom}(\hat{H})\) is not NP-complete. Let the Hamiltonian cycle be \(v_0, v_1, \ldots, v_n-1, v_0\) and without loss of generality all edges are blue with the possible exception of \(v_0v_1\). Consider \(H - v_0\). This is switching equivalent to a segmented signed graph with Hamiltonian path \(v_1, v_2, \ldots, v_{n-1}\) (where \(n\) is even), and thus it has the structure described above.

By symmetry, there is a right-leaning segment. Let \(v_i\) be the first vertex of the first right-leaning segment. Then \(v_{i+1}\) has degree 2 in \(H - v_0\) and degree 2 in \(H\) unless \(v_0v_{i+1}\) is a bicoloured edge. If \(v_0v_{i+1}\) is an edge, then \(i + 1\) is odd and the forward source \(v_i\) sends a
bicoloured edge to each \( v_0 \) for \( o \) odd with \( i + 3 \leq o \leq n - 1 \). Consequently, \( v_0v_{i+1}v_iv_{n-1}v_0 \) is an alternating 4-cycle. We conclude \( v_{i+1} \) has degree 2.

Rename the vertices of the underlying graph \( H \) so that \( v_0 \) is a vertex of degree two. The signed graph \( \hat{H} - v_0 \) is path separable, with the spanning path \( v_1, v_2, \ldots, v_{n-1} \) of unicoloured edges being all blue, while possibly one of the edges \( v_0v_1, v_0v_{n-1} \) is red (and the other one is blue). Again we assume that \( \text{List-S-Hom}(\hat{H} - v_0) \) is not NP-complete, so Theorem 29 implies that \( \hat{H} - v_0 \) is switching equivalent to a segmented signed graph. By symmetry, we may assume \( v_2 \) is adjacent to \( v_{n-1} \) (and hence \( n \) is even), otherwise \( \hat{H} - v_0 \) contains an induced cycle of length greater than four. Now we have a 4-cycle \( v_0, v_1, v_2, v_{n-1}, v_0 \).

Assume first that \( v_2, v_3, v_4, v_5, v_2 \) is also an edge, else would have 4-cycle pair. By repeated use of the five-path \( P_5 \) with middle edge bicoloured (case (b) of family \( F \)), we conclude that \( v_6v_{n-1}, v_8v_{n-1}, \ldots, v_{n-4}v_{n-1} \) are also edges. Similarly, \( v_2v_7, v_2v_9, \ldots, v_2v_{n-1} \) must also be edges. Thus using our descriptions of the polynomial path-separable cases, we conclude that \( \hat{H} - v_0 - v_1 = v_2, v_3, \ldots, v_{n-1} \) is just one segment. If \( v_{n-1}, v_{n-2}, v_{n-3}, v_{n-4} \) is a 4-cycle, the argument is similar.

If neither \( v_2, v_3, v_4, v_5, v_2 \) nor \( v_{n-1}, v_{n-2}, v_{n-3}, v_{n-4} \) is a 4-cycle, then from Theorem 29 we conclude that \( \hat{H} - v_0 \) is left-right-segmented, with a left-right-leaning segment \( S \) not at the end of \( P \). This is easy to dismiss, because there would be a \( P_3 \) involving the segment \( S \) and the vertex \( v_0 \). In conclusion \( \hat{H} - v_0 - v_1 \) is just one segment.

If the cycle \( C \) is balanced, this conclusion means that \( \hat{H} \) is switching equivalent to some \( \hat{H}_\ell \) with \( \ell \geq 3 \). If the cycle \( C \) is unbalanced with \( n = 6 \), then this means \( \hat{H} \) is switching equivalent to \( \hat{H}_1 \) and we prove that otherwise the problem is NP-complete. Therefore, assume that \( n > 6 \) and \( C \) is unbalanced; without loss of generality, suppose that the edge \( v_0v_1 \) is red.

We will reduce from one of the NP-complete cases of Boolean satisfiability dichotomy theorem of Schaefer [24].

Consider Boolean variables \( V \) and a set of quadruples \( R \) on these variables. An instance of the problem asks for an assignment in which every quadruple \( (a', b', c', d') \in R, (a' = b' = c' = d') \vee (a' \neq c') \).

Schaefer [24] proved that a Boolean constraint satisfaction problem is NP-complete except for the well-known polynomial cases of 2-SAT, Horn clauses, co-Horn clauses, linear equations modulo two, or when the only satisfying assignments are the all true or the all false assignments. To see that our problem is not expressible as 2-SAT, consider the following three satisfying assignments for \( (a', b', c', d') : (1, 1, 1, 1), (0, 0, 0, 0), (0, 0, 1, 0) \). It is well known, see e.g. [9] Lemma 4.9, that any problem expressible as 2-SAT has the property that the majority function on three satisfying assignments must also be a satisfying assignment. However, for our three assignments the majority function yields the assignment \( (1, 0, 1, 0) \) which is not satisfying. Similarly, our problem is not expressible as Horn clauses (respectively co-Horn clauses) because the minimum (respectively maximum) function on the two satisfying assignments \( (0, 1, 1, 0), (1, 1, 0, 0) \) is not a satisfying assignment, cf. Lemma 4.8 in [9]. Finally, our problem is not expressible by linear equations modulo two because the sum modulo two of the three satisfying assignments \( (1, 1, 1, 1), (1, 1, 0, 1), (0, 1, 1, 1) \) results in the assignment \( (0, 1, 0, 1) \) which is not satisfying, cf. Lemma 4.10 in [9]. Thus our problem is one of the NP-complete cases.

Suppose that we have been given a set of quadruples \( R \) on a set \( V \) of Boolean variables. We shall now construct a corresponding signed graph \( \hat{G} \).

For each quadruple \( (a', b', c', d') \), we construct a quadruple gadget \( Q(a', b', c', d') \) with lists as in the Figure 18. The repetition of some vertices in lists in the gadget ensures that each longest path can be mapped to a walk of length \( k + 1 \) in \( \hat{H} \) between \( b \) and \( w \). If there is
some element \( r \) contained in the first or second position of two or more quadruples, then we add a new vertex \( x_r \) and a blue edge from \( x_r \) to each occurrence of \( r \). The analogous argument can be used if an element is contained in the third or fourth position of two or more quadruples. Furthermore, if there is some element \( r \) in the first or second position of a quadruple and also in the third or fourth position of another quadruple, then we add a blue path \( r_1, \ldots, r_{k-2} \) with \( L(r_i) = \{ t_i \} \) and connect each endpoint with one occurrence by a blue edge. Denote the resulting graph by \( \hat{G} \).

We claim that there exists a satisfying assignment for \( R \) if and only if there is a list homomorphism from \( \hat{G} \) to \( \hat{H} \).

For any homomorphism \( f: \hat{G} \to \hat{H} \), we define an assignment \( \pi_f \) in which a variable \( x \) is 1 if the vertex \( x \) was switched in the homomorphism \( f \) and is 0 otherwise. It can be checked that in any homomorphism \( f \), all occurrences of a given variable must be switched or non-switched, because of the mutual connections, thus this is a well defined correspondence.

Furthermore, for a given quadruple gadget, its inner vertices must all choose either the first element described in its list or the second in every possible list homomorphism.

The last thing we need to argue is that certain switching of the endpoints of the quadruple gadget are not possible and some of them are possible. We denote a particular switching as elements from lists of the quadruple gadget. We divide the analysis of possible switchings into two cases, based on the choice of elements from lists of the quadruple gadget.

- **The first elements of lists are being chosen.** Under this mapping, we have to make sure that \( P(a', c') \) is positive. As this path is negative without any switching, we need to switch either at \( a' \) and not in \( c' \) or vice versa to have this path positive. Switching at \( b' \) or \( d' \) does not affect this situation. Hence the only admissible quadruples in this case are the ones where the first and the third coordinates differ.

- **The second elements of lists are being chosen.** In this case, all main paths have to map to a negative walk. This is true if we are in case \((0,0,0,0)\) and subsequently in case \((1,1,1,1)\) as in that case, the sign of all main paths remains the same as if we do not switch anywhere. Whenever we have a quadruple with two coordinates having different values, the corresponding path has to be positive and thus we arrive into problems. Hence \((0,0,0,0)\) and \((1,1,1,1)\) are the only possible quadruples for this case.

The argument is concluded by observing that every satisfying assignment corresponds to a
list homomorphism with the respective occurrences switched or non-switched, and vice versa.

**Polynomial cases.**

It remains to show that all remaining cycle-separable graphs can be solved by a polynomial algorithm.

If \( \hat{H} \) is switching equivalent to \( \hat{H}_0 \), then the problem \textsc{List-S-Hom}(\( \hat{H} \)) is polynomial-time solvable by Theorem 6.

Let \( \hat{G} \) be a connected bipartite graph. We will call the vertices of parts of bipartition in both \( G \) and \( H \) as black and white, respectively. First, we try mapping the black vertices of \( G \) to the black vertices of \( H \). If that fails, we try to map the white vertices of \( G \) to the black vertices \( \hat{H} \). Suppose that the shorter segment has consecutively vertices \( b = s_0, s_1, s_2, s_3 = w \) (called \( s \)-vertices) and \( b \) is black and \( w \) is white. The vertices of the longer segment will be denoted as \( b = t_0, t_1, \ldots, t_k = w \) (and called \( t \)-vertices).

We order the vertices of two segments \( \hat{H} \) by a pair \(<_b,<_w\), where \(<_b\) is a linear ordering of the black vertices and \(<_w\) is a linear ordering of the white vertices in the following way. For black vertices, we define \( b = s_0 <_b s_1 <_b s_2 <_b t_0 <_b t_2 \ldots <_b t_k \) and for white vertices, \( w = s_3 <_w s_1 <_w t_k <_w t_{k-2} <_w \ldots <_w t_1 \).

First, we perform arc consistency procedure and also bicoloured arc consistency procedure. If there is a vertex with empty list after this step, then no suitable list homomorphism exists.

Otherwise, we define two mappings \( f_1 \) and \( f_2 \) as

\[
\begin{align*}
& f_1(v) := t_i \in L(v) \text{ with the smallest index } i, \\
& f_2(v) := s_i \in L(v) \text{ with the smallest index } i.
\end{align*}
\]

Let \( u \) be a black vertex and \( v \) be a white vertex of \( \hat{G} \) and suppose there is a bicoloured edge between a vertex from \( L(u) \) and \( L(v) \). Clearly it has the form \((2i)(2j + 3)\), where \( i \geq j \). Since \( f_1(v) \) the minimum of the list \( L(u) \), it holds that \( f_1(u) \leq 2i \) and, analogously, since \( f_1(v) \) is the maximum of \( L(v) \), it holds \( f_1(v) \geq 2j + 3 \). This implies \( f_1(v) \geq f_1(u) + 3 \) and consequently, \( f_1(u)f_2(v) \) is a bicoloured edge. The same applies for \( f_2 \). We conclude that both \( f_1 \) and \( f_2 \) are list homomorphisms from \( \hat{G} \) to \( \hat{H} \).

If \( b \in L(v) \) for some black vertex \( v \), then \( f_1(v) = f_2(v) = b \). Analogously, if \( w \in L(v) \) for some white vertex \( v \), then \( f_1(v) = f_2(v) = w \). Observe that \( t \) dominates all black vertices and \( w \) dominates all white vertices. The vertices with \( b \) or \( w \) is its image will be called \textit{boundary vertices}. The rest of \( G \) breaks into \textit{regions} (defined in a similar way as in Section 6). For each region, all its vertices map either to \( s \)-vertices or all to \( t \)-vertices.

First suppose that \( \hat{H} \) is switching equivalent to \( \hat{H}_s \) with odd \( s \geq 3 \). Let \( K \) be some region of \( G \). If the lists of vertices of \( K \) contain only \( s \)-vertices or only \( t \)-vertices, then there is no choice and we will use mappings \( f_2 \) or \( f_1 \), respectively. Now suppose that the lists of vertices of \( K \) contains both \( s \)-vertices and \( t \)-vertices. If there is a homomorphism which maps vertices of \( K \) to \( s \)-vertices (using \( f_1 \)), then it maps edges in \( K \) to blue edges (since there are no bicoloured edges). On the other hand, if there is a homomorphism which maps vertices of \( K \) to \( t \)-vertices (using \( f_2 \)), then it maps edges in \( K \) to blue or bicoloured edges. Thus in that case, we can always use mapping \( f_2 \). This gives us homomorphism which maps bicoloured edges to bicoloured edges (since we did bicoloured arc consistency test) and it remains to consider the signs of cycles which were mapped to unicoloured edges. That can be done through solving a suitable instance of 2-SAT which ensures that the boundary points are switched so that any walk through a given region between any two boundary points is negative. This is again analogous to the procedure given in Section 5 on polynomial algorithm for irreflexive signed trees. This completes the analysis of this case.

Suppose that \( \hat{H} \) is switching equivalent to \( \hat{H}_1 \). We need to decide for each region whether it will be mapped to the \( s \)-vertices or \( t \)-vertices and whether to switch or not at vertices.
mapped to $b$ and $w$.

Let $K$ be a region. (We can think of $K$ as being a single edge as its image must be and we can switch so that it is all blue or red). We now have boundary vertices $v_1, v_2, \ldots$ mapping to $b$ and $u_1, u_2, \ldots$ mapping to $w$. We need to switch the boundary vertices so that the region maps to the $t$-vertices or the $s$-vertices.

Note that under any suitable homomorphism, the input graph $\hat{G}$ must be switched so that walks from $v_i$ to $v_j$ must be positive and similarly for $u_i$ to $u_j$. To code this, for each vertex $v_i$, we will introduce a variable $x_i$, and similarly for $u_j$, we introduce a variable $y_j$.

If there is a positive walk from $v_1$ to $v_2$, we add $x_1 = x_2$ (and so on for all pairs from $x_1, x_2, \ldots$). If there is a negative walk, we add $x_1 \neq x_2$. Similarly for $u_1, u_2, \ldots$.

Finally, we need equations to code the sign of walks from $v_i$ to $u_j$. If $K$ maps to the $t$-vertices, then walks between $v_i$ and $u_j$ must be positive. We can code the needed switching with $x_i = i_j$ or $x_i \neq u_j$. Similarly, for $K$ mapping to the $s$-vertices, we must make all walks between $v_i$ and $y_j$ negative. If $K$ has a choice, then we can use a variable $z_k$ that is 0 if $K$ maps to $t$-vertices and 1 if $K$ maps to $s$-vertices. For this situation, we shall add equations $x_i = y_i + z_k$.

\section{Conclusions}

It seems difficult to give a full combinatorial classification of the complexity of list homomorphism problems for general signed graphs. We have accomplished this for the special case of signed trees with possible loops. The polynomial algorithms rely on min ordering or on majority polymorphisms, and neither of the methods alone is sufficient. We also obtain a full complexity classification for the special case of irreflexive signed graphs where the unicoloured edges form a spanning path or cycle.

In a complementary development, a recent manuscript of Kim and Siggers \cite{kim2021} gives an algebraic characterization of signed graphs for which the list homomorphism problem is polynomial time solvable, and proposes a possible combinatorial characterization of irreflexive signed graphs with this property. The manuscript also nicely highlights the central importance of irreflexive signed graphs for the list homomorphism problems.

\section*{References}

1. Jørgen Bang-Jensen, Pavol Hell, and Gary MacGillivray. The complexity of colouring by semicomplete digraphs. \textit{SIAM J. Discrete Math.}, 1(3):281–298, 1988.
2. Jan Bok, Richard Brewster, Tomáš Feder, Pavol Hell, and Nikola Jedličková. List homomorphism problems for signed graphs. In Javier Esparza and Daniel Kráľ, editors, 45th International Symposium on Mathematical Foundations of Computer Science (MFCS 2020), volume 170 of \textit{Leibniz International Proceedings in Informatics (LIPIcs)}, pages 20:1–20:14, Dagstuhl, Germany, 2020. Schloss Dagstuhl–Leibniz-Zentrum für Informatik. URL: \url{https://drops.dagstuhl.de/opus/volltexte/2020/12688}, doi:10.4230/LIPIcs.MFCS.2020.20.
3. Jan Bok, Richard C. Brewster, Pavol Hell, and Nikola Jedličková. List homomorphisms of signed graphs. In Bordeaux Graph Workshop, pages 81–84, 2019.
4. Richard C. Brewster. \textit{Vertex colourings of edge-coloured graphs}. ProQuest LLC, Ann Arbor, MI, 1993. Thesis (Ph.D.)–Simon Fraser University (Canada).
5. Richard C. Brewster, Florent Foucaud, Pavol Hell, and Reza Naserasr. The complexity of signed graph and edge-coloured graph homomorphisms. \textit{Discrete Mathematics}, 340(2):223–235, 2017.
6. Richard C. Brewster and Timothy Graves. Edge-switching homomorphisms of edge-coloured graphs. \textit{Discrete Math.}, 309(18):5540–5546, 2009.
Richard C. Brewster and Mark Siggers. A complexity dichotomy for signed $h$-colouring. *Discrete Mathematics*, 341(10):2768–2773, 2018.

Andrei A. Bulatov. A dichotomy theorem for nonuniform CSPs. In *58th Annual IEEE Symposium on Foundations of Computer Science—FOCS 2017*, pages 319–330. IEEE Computer Soc., Los Alamitos, CA, 2017.

N. Creignou, S. Khanna, and Sudan M. Complexity classifications of boolean constraint satisfaction problems. In *SIAM Monographs on Discrete Mathematics and Applications*, 2001.

Tomás Feder and Pavol Hell. List homomorphisms to reflexive graphs. *Journal of Combinatorial Theory, Series B*, 72(2):236–250, 1998.

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

Tomás Feder, Pavol Hell, Peter Johnsson, Andrei Krokhin, and Gustav Nordh. Retractions to pseudoforests. *SIAM J. Discrete Math.*, 24(1):101–112, 2010.

Tomás Feder and Moshe Y. Vardi. The computational structure of monotone monadic SNP and constraint satisfaction: a study through Datalog and group theory. In *STOC*, pages 612–622, 1993.

Florent Foucaud and Reza Naserasr. The complexity of homomorphisms of signed graphs and signed constraint satisfaction. In *Latin American Symposium on Theoretical Informatics*, pages 526–537. Springer, 2014.

Bertrand Guenin. Packing odd circuit covers: A conjecture. Manuscript, 2005.

Frank Harary. On the notion of balance of a signed graph. *Michigan Math. J.*, 2:143–146 (1955), 1953/54.

Frank Harary and Jerald A. Kabell. A simple algorithm to detect balance in signed graphs. *Math. Social Sci.*, 1(1):131–136, 1980/81.

Pavel Hell and Jaroslav Nešetřil. On the complexity of $H$-coloring. *J. Combin. Theory Ser. B*, 48(1):92–110, 1990.

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

Peter Jeavons. On the algebraic structure of combinatorial problems. *Theoret. Comput. Sci.*, 200(1-2):185–204, 1998.

Hyobin Kim and Mark Siggers. Towards a dichotomy for the switch list homomorphism problem for signed graphs. Manuscript, 2021.

Reza Naserasr, Edita Rollová, and Éric Sopena. Homomorphisms of signed graphs. *J. Graph Theory*, 79(3):178–212, 2015.

Reza Naserasr, Éric Sopena, and Thomas Zaslavsky. Homomorphisms of signed graphs: An update, 2019. [arXiv:1909.05982](https://arxiv.org/abs/1909.05982).

Thomas J Schaefer. The complexity of satisfiability problems. In *Proceedings of the tenth annual ACM symposium on Theory of computing*, pages 216–226, 1978.

Thomas Zaslavsky. Characterizations of signed graphs. *J. Graph Theory*, 5(4):401–406, 1981.

Thomas Zaslavsky. Signed graph coloring. *Discrete Math.*, 39(2):215–228, 1982.

Thomas Zaslavsky. Signed graphs. *Discrete Appl. Math.*, 4(1):47–74, 1982.

Thomas Zaslavsky. Is there a matroid theory of signed graph embedding? *Ars Combin.*, 45:129–141, 1997.

Thomas Zaslavsky. A mathematical bibliography of signed and gain graphs and allied areas. *Electron. J. Combin.*, 5:Dynamic Surveys 8, 124, 1998. Manuscript prepared with Marge Pratt.

Dmitriy Zhuk. A proof of CSP dichotomy conjecture. In *58th Annual IEEE Symposium on Foundations of Computer Science—FOCS 2017*, pages 331–342. IEEE Computer Soc., Los Alamitos, CA, 2017.