Clique-width: When Hard Does Not Mean Impossible*

Robert Ganian[1], Petr Hliněný[1], and Jan Obdržálek[1]

1 **Masaryk University, Brno**
`ganian,hlineny,obdrzalek@fi.muni.cz`

Abstract

In recent years, the parameterized complexity approach has lead to the introduction of many new algorithms and frameworks on graphs and digraphs of bounded clique-width (or, equivalently, rank-width). However, despite intensive work on the subject, there still exist well-established hard problems where neither a parameterized algorithm nor a theoretical obstacle to its existence are known. Our article is interested mainly in the digraph case, targeting the well-known Minimum Leaf Out-Branching (cf. also Minimum Leaf Spanning Tree) and Edge Disjoint Paths problems on digraphs of bounded clique-width with highly non-standard new approaches.

In the first part – for the Minimum Leaf Out-Branching problem, which has remained open till now despite attempting various traditional approaches based on dynamic programming – we design an inspiring novel XP-time algorithm wrt. clique-width. We remark that this problem is known to be W[2]-hard, and that our algorithm is very different from previously published attempts solving special cases of it such as the Hamiltonian Path. The second part then looks at the Edge Disjoint Paths problem (both on graphs and digraphs) from an unexpected perspective – rather surprisingly showing that this problem has an involved definition in the MSO_1 logic of graphs, i.e. a logic not capable of quantification over edges. A linear-time FPT algorithm for Edge Disjoint Paths wrt. clique-width is then a straightforward consequence.

1998 ACM Subject Classification Algorithms and data structures, computational complexity

Keywords and phrases clique-width, bi-rank-width, minimum leaf out-branching, minimum leaf spanning tree, edge-disjoint paths

1 Introduction

It is known that the majority of graph problems are NP-complete in general, so alternative approaches are necessary for tackling these problems. The utilization of parameterized algorithmics is one such very successful approach, where instead of focusing on the general class of all graphs we design algorithms on graphs with a bounded structural parameter (or "width"). This has strong practical motivation, since real-world applications generally work with specific classes of graphs as input.

"Polynomial runtime" parameterized algorithms are roughly divided into two groups. The more ideal case constitutes *fixed-parameter tractable* (FPT) algorithms, where the runtime is $poly(n) \cdot f(k)$ (n being the input size and k the parameter). Unfortunately, not all combinations of problems and parameters allow FPT algorithms, and so in some cases it is necessary to settle for an *XP algorithm* – i.e. an algorithm with runtime $poly(n)^{f(k)}$. Notice that the exponent in XP algorithms increases with the parameter, but still the runtime remains polynomial for any fixed value of k.

* This work was supported by the Czech research grant GAČR 201/09/J021.

As for the parameters themselves, the one best known today is the tree-width of Robertson and Seymour [17] which has allowed for efficient solution of many NP-hard problems on all graphs having bounded tree-width. The drawback is that the class of graphs with bounded tree-width is quite restrictive. A lot of research since then has focused on obtaining a width measure which would be more general and still allow efficient algorithms for a wide range of NP-hard problems on graphs of bounded width. This has lead to the introduction of clique-width by Courcelle and Olariu [4] and, subsequently, of rank-width by Oum and Seymour [16]. Both of these width parameters are related in the sense that one is bounded if and only if the other is bounded. We refer to Section 2 for further details.

In this article, we provide polynomial algorithms for two well-established problems on digraphs of bounded clique-width/bi-rank-width.

- The first one is *Minimum Leaf Out-Branching*, a problem which generalizes the Hamiltonian Path problem and which is studied e.g. in [5]. The task is to find a spanning out-tree in a digraph that minimizes the number of leaves. Definition and more details are provided in Section 3. We remark that the undirected variant is known as Minimum Leaf Spanning Tree problem (e.g. [19]), and our results apply also to that.
- The second one is *Edge Disjoint Paths* problem, asking for pairwise edge-disjoint paths between a fixed number of terminal pairs. In this case the directed variant is much more difficult than the undirected one – see details in Section 4. Again, disjoint paths problems are somehow related to Hamiltonian Path.

Parameterized complexity status of Minimum Leaf Out-Branching remained unsolved in our previous work on digraphs of bounded bi-rank-width [9], resisting the dynamic programming approaches traditionally used e.g. for clique-width. The provided new Algorithm 12 solves the problem and is also straightforwardly applicable to undirected graphs.

▶ **Theorem 1** (Algorithm 12). *The Minimum Leaf Out-Branching problem on a given digraph G of clique-width k (with arbitrary number of leaves) can be solved in XP time $\mathcal{O}(n^{f(k)})$, where $f(k) \sim 2^{\mathcal{O}(k)}$ if a k-expression of G is given, and $f(k) \sim 2^{\mathcal{O}(2^k)}$ otherwise.*

The second part of the article deals with the Edge Disjoint Paths problem with a fixed number of paths. Note that this was the only remaining open (directed) variant of Disjoint Paths with respect to parameterization by clique-width – see [9, 12, 15] for complexity results and/or algorithms for the other variants. Somehow surprisingly, we show that even the Edge Disjoint Paths problem may actually be described by an MSO_1 formula, and so is solvable in linear FPT time on digraphs of bounded clique-width [3]. However, unlike most known problems expressible in MSO_1, the formula for Edge Disjoint Paths is quite intricate and Section 4 contains a description of the construction of the formula as well as the proof that it exactly captures the Edge Disjoint Paths problem. In the end we obtain:

▶ **Theorem 2** (Theorem 17). *Both the undirected and directed variant of the Edge Disjoint Paths problem with a fixed number of terminal pairs have a linear-time FPT algorithm on simple (di)graphs of bounded clique-width.*

Theorem 2 can, moreover, be directly used as a subroutine in a new algorithm for the Edge Disjoint Paths problem on tournaments by Chudnovsky and Seymour [in preparation].

2 Clique-width and rank-width

In the article we use standard graph and digraph (directed graph) notation. All our graphs and digraphs are simple (i.e. do not contain loops or multiple edges) unless specified otherwise.

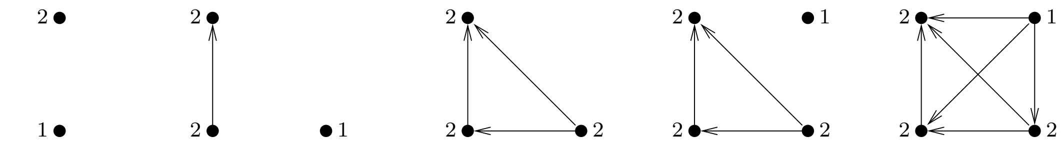
▶ **Definition 3** (clique-width, [4]). Let k be a positive integer. A pair (G, γ) is a *k-labelled graph* if G is a graph and $\gamma : V(G) \to \{1, 2, \ldots, k\}$ is a mapping. We call $\gamma(v)$ for $v \in V(G)$ the *label* of a vertex v. As γ is usually fixed, we often write just G for the k-labelled graph (G, γ), and we refer to $\gamma(v)$ as to the G-label of a vertex v. A *k-expression* is a well formed expression built using the four operators defined below. Let $1 \le i, j \le k$. Then

1. $[i]$ is a nullary operator which represents a graph with a single vertex labelled i,
2. $\eta_{i,j}$, for $i \neq j$, is a unary operator which adds edges between all pairs of vertices where one is labelled i and the other is labelled j,
3. $\rho_{i \to j}$ is a unary operator which changes the labels of all vertices labelled i to j, and
4. $\oplus$ is a binary operator which represents disjoint union of two k-labelled graphs.

Each k-expression naturally corresponds to (generates) a k-labelled graph G. The *clique-width* of an undirected graph G is then the smallest k such that there exists a k-expression generating G. For digraphs clique-width is defined in just the same way, only the operator $\eta_{i,j}$ is replaced by the operator $\alpha_{i,j}$ which creates directed edges (arcs) from each vertex with label i to each vertex with label j. It is known [4] that every graph of clique-width k can be generated by an *irredundand* expression, i.e. an expression that applies the $\eta_{i,j}$ / $\alpha_{i,j}$ operator only in situations when there is no edge from a vertex of label i to one of label j.

It is quite natural to view a k-expression t_G corresponding to G as a tree T with nodes labelled by subterms of t_G (t_G being the root), together with a bijection between the leaves of the tree and vertices of G. In this setting the *type* of each node $t \in V(T)$ is the top-level operator of t, so we have four different node types. We will also use the following notation: For a node $t \in V(T)$ let $G[t]$ be the subgraph of G given by t.

▶ **Example 4.** $\alpha_{1,2}(\rho_{1\to2}(\alpha_{1,2}(\rho_{1\to2}(\alpha_{1,2}([1]\oplus[2]))\oplus[1]))\oplus[1])$ is a 2-expression corresponding to a directed clique of size 4. See Fig. 1.



$A = [1] \oplus [2]$ $B = \rho_{1\to2}(\alpha_{1,2}(A)) \oplus [1]$ $C = \rho_{1\to2}(\alpha_{1,2}(B))$ $D = C \oplus [1]$ $E = \alpha_{1,2}(D)$

Figure 1 Construction of the directed clique of size 4

Closely related to clique-width is another structural parameter, called *rank-width* [16] (on undirected graphs) or *bi-rank-width* [14] (on digraphs). Due to space restrictions we only refer to [11] for their definitions. The relationship of these measures to the former is that they are bounded if and only if clique-width is bounded. However, a crucial advantage of rank-width is that it can be computed optimally by an FPT algorithm. To be more specific:

▶ **Theorem 5** ([2, 16]). $rwd(G) \le cwd(G) \le 2^{rwd(G)+1} - 1$ *for all graphs* G.

▶ **Theorem 6** ([13, 14]). *For every integer parameter t there is an $O(n^3)$-time FPT algorithm that, for a given n-vertex graph G, either finds a bi-rank-decomposition of G of width at most t, or confirms that the bi-rank-width of G is more than t.*

Due to lack of space for comprehensible definitions and explanation of rank-width and their parse trees we stick with (perhaps better known) clique-width in this article. All the results, however, could be straightforwardly reformulated for (bi-)rank-width.

3 Minimum leaf out-branching

Let $out_G(x)$ denote the out-degree of x in a digraph G, i.e. the number of edges having their tail in x. For an edge f and nonadjacent vertices x, y of a digraph G, we write $G - f$ to denote the graph resulting by removal f from G, and $G + (x, y)$ for the graph obtained by adding a new edge from x to y. A digraph T is an *out-tree* if T is an oriented tree with only one vertex of in-degree zero (called the *root*). The vertices of out-degree zero are called *leaves* of T. An *out-forest* is a digraph whose weakly connected components are out-trees.

▶ **Definition 7.** Let G be a digraph. We say that T is an *out-branching* of G if T is a spanning subdigraph of G, i.e. $V(T) = V(G)$ and $E(T) \subseteq E(G)$, and T is an out-tree. The *Minimum Leaf Out-Branching problem* (or MinLOB for short) is the problem of deciding, for a digraph G and integer ℓ on the input, whether G contains an out-branching with at most ℓ leaves.

Notice that not every digraph has an out-branching. It is not hard to show that G has an out-branching if, and only if, there is a vertex $v \in V(G)$ such that there is a directed path from v to any vertex of G. This is checkable in linear-time [1], but the MinLOB problem itself is NP-complete since it contains the Hamiltonian Path as a special case ($\ell = 1$). It is also possible to analogically define the *Maximum Leaf Out-Branching problem* (MaxLOB), asking for an out-branching with at least ℓ leaves, but this variant seems to have quite different (and rather easier) algorithmic behaviour than MinLOB.

The core contribution of our paper is to resolve the important open question of [9]; what is the computational complexity of MinLOB when parameterized by the clique-width / bi-rank-width of the input graph? It follows by a reduction from Hamiltonian Path [7] that MinLOB is W[2]-hard with respect to clique-width (even with fixed ℓ), and so does not have an FPT algorithm unless the Exponential Time Hypothesis fails. Historically the first XP algorithm for an undirected Hamiltonian Path parameterized by clique-width was due to Espelage et al [6]. An XP algorithm for ℓ-MinLOB for every fixed ℓ and parameterized by the bi-rank-width was given in [11], but its approach could not be extended towards the general case of ℓ on the input.

We consider MinLOB conceptually important in the area of parameterized algorithms since none of the established design approaches (e.g. with dynamic programming) seems to apply to this case. Yet, we have managed to come with a new involved approach that (we think) deserves further investigation, too.

3.1 Out-branching and modules

We first show some basic properties of the problem as a prelude to the coming algorithm in the next section. Though these properties (cf. Definitions 8, 9) are not directly used in Algorithm 12 and its proof, we consider them worth independent interest.

For a digraph G, a set $M \subseteq V(G)$ is called a *module* if every vertex of M has the same in-neighbourhood and out-neighbourhood (as every other in M) among the vertices not in M. Generalizing the module concept, we consider a k-labelled digraph (H, γ) such that $H \subseteq G$. We say that H is a *labelled-modular subdigraph* of G if $\gamma^{-1}(i)$ is a module in $G - E(H)$ for all $i = 1, 2, \ldots, k$. Note that H is not required to be an induced subdigraph of G.

In other words, H is a labelled-modular subdigraph of G if the existence of an edge in $G - E(H)$ incident with some $v \in V(H)$ "depends only on" the label of v. Notice that if s is a subexpression of a (irredundand) k-expression t, then the generated k-labelled digraph

$G[s]$ is always a labelled-modular subdigraph of the whole $G[t]$ (an analogical claim holds e.g. for bi-rank-decompositions).

Let G be a digraph, $H \subseteq G$ its subgraph and $F \subseteq H$ an out-forest. We call the pair (F, μ) where $\mu : V(H) \to \mathbb{N}$ an *annotated out-forest*. We say that the annotated out-forest (F, μ) *extends* to an out-branching $T \subseteq G$ if $E(F) = E(T) \cap E(H)$, and for all $x \in V(H)$ we have $\mu(x) = out_T(x) - out_F(x)$. We are going to define an equivalence relation $\approx_H$ on the set of all annotated out-forests of a k-labelled graph H, with the intended meaning to "capture all important information" about possible extendability of a particular annotated out-forest into an out-branching.

▶ **Definition 8** (Canonical equivalence)**.** A pair of annotated out-forests (F_1, μ_1) and (F_2, μ_2) in a k-labelled digraph H is canonically equivalent, written as $(F_1, \mu_1) \approx_H (F_2, \mu_2)$, if, and only if, the following holds for each integer ℓ and every digraph G such that H is a labelled-modular subdigraph of G: (F_1, μ_1) can be extended to an out-branching of G with $\leq \ell$ leaves if and only if (F_2, μ_2) can be extended to an out-branching of G with $\leq \ell$ leaves.

On the other hand, in Definition 9 we introduce simple "information about (F, μ)" that is sufficient to determine its equivalence class within $\approx_H$. For every connected component (out-tree) T_0 of F in H, including the isolated vertices of H not incident with any edge of F, the *shape* of T_0 is the pair (a, B) where a is the H-label of the root of T_0 and B is the set of all H-labels occurring at the vertices $x \in V(T_0)$ such that $\mu(x) > 0$ (*active* vertices).

▶ **Definition 9** (Out-forest signatures)**.** The *signature* of a (spanning) annotated out-forest (F, μ) in a k-labelled digraph H is a vector in $\mathbb{N}^*$ consisting of

- the number of leaves x of F (incl. isolated vertices) such that $\mu(x) = 0$,
- for every $i = 1, \ldots, k$, the sum of $\mu(x)$ over all vertices $x \in V(H)$ of the H-label i, and
- for every possible shape, the number of out-trees of F having this shape.

Notice that the length of this vector depends only on k and not on the size of H.

▶ **Lemma 10.** *Let (F_1, μ_1) and (F_2, μ_2) be a pair of annotated out-forests in a k-labelled digraph H. If the signatures of (F_1, μ_1) and (F_2, μ_2) are equal, then $(F_1, \mu_1) \approx_H (F_2, \mu_2)$.*

Due to lack of space, we skip the proof of this lemma (see the Appendix). The claim clearly suggests that an XP-time algorithm for MinLOB might exist since the information "carried by" the set of available signatures is of polynomial size. Unfortunately, even this strong claim is not strong enough to give such an algorithm (unlike in the finite Myhill–Nerode-type case, e.g. [8], or in many other XP solvable problems [11]) since we do not know how to process available signature vectors dynamically along a k-expression.

3.2 A dynamic algorithm for MinLOB

In order to obtain an XP algorithm for the MinLOB problem, we introduce a "weaker" alternative to Definition 9. Recall that a vertex x of an annotated out-forest (F, μ) is *active* if $\mu(x) > 0$. We now relax this notion to suit the coming algorithm.

Assume a k-expression t generating the digraph $H = G[t]$, an annotated out-forest (F, μ) in H, and a vertex $v_q \in V(H)$ generated by the leaf q of t. We say that v_q is *potentially active* in H for the k-expression t if, for every node (subexpression) s of t on the path from q to the root, the annotated out-forest induced by (F, μ) in $G[s] \subseteq H$ contains an active vertex (possibly v_q itself) of the same $G[s]$-label as that of v_q. In particular, if a vertex x is active in (F, μ), then x is also potentially active for t. If there is no active vertex of label i

in (F, μ), then there is also no such potentially active vertex. It may, however, happen that there are many more potentially active vertices of (F, μ) for t than the active ones.

For every connected component (out-tree) T_0 of F in H, we define the *weak shape* of T_0 as the pair (a, B) where a is the H-label of the root of T_0 and B is the set of all H-labels occurring at the vertices $x \in V(T_0)$ that are potentially active for the k-expression t.

▶ **Definition 11** (Weak signature, cf. Definition 9)**.** The *weak signature* of a spanning annotated out-forest (F, μ) in the k-labelled digraph $H = G[t]$ generated by a k-expression t is a vector $\vec{w}$ in $\mathbb{N}^c$ (with the appropriate length c) consisting of the sections

- wl, the number of leaves x of F (incl. isolated vertices) such that $\mu(x) = 0$,
- $wa(i)$ for every $i = 1, \dots, k$, where $wa(i)$ equals the sum of $\mu(x)$ over all vertices x of the H-label i (informally, the "total multiplicity" of all active vertices of label i), and
- $ws(a, B)$ for every possible weak shape (a, B), equal to the number of out-trees (weak components) of F having this weak shape (a, B) in H for the k-expression t.

The advantage of a weak signature over former signature is that weak signatures are easier to handle in dynamic programming on a k-expression of the input graph. Still, the situation is not as easy as if we could dynamically compute the set of all weak signatures of all possible annotated outforests in our graph — we can only compute a suitable superset of it via the following straightforward algorithm.

▶ **Algorithm 12.** Assume an input consisting of a k-expression t generating a k-labelled digraph G on n vertices. The following algorithm computes, in XP-time wrt. the parameter k, a set $\mathcal{U}$ of vectors from $\mathbb{N}^c$ (cf. Definition 11) such that $\mathcal{U}$ includes all weak signatures of spanning annotated out-forests in G for t.

I. Input t (a k-expression); $G = G[t]$.

II. At every leaf $q = [i]$ of t (where $i \in \{1, \dots, k\}$), the graph $G[q]$ is actually a single vertex v_q of label i. Let U_q be the set of weak signatures of the edge-less annotated out-forests $(G[q], \mu_j)$ where $\mu_j(v_q) = j$, over $0 \le j \le out_G(v_q)$.

III. At every internal node r of t, we compute in the leaves-to-root direction as follows.

$r = p \oplus q$**:** U_r is the set, for all pairs $\vec{c} \in U_p$, $\vec{d} \in U_q$, of their vector sums $\vec{c} + \vec{d}$.

$r = \rho_{i \to j}(q)$**:** We initialize $U_r = \emptyset$. Then, for every $\vec{c} \in U_q$, $\vec{c} = (wl, \vec{wa}, \vec{ws})$ as in Definition 11, we compute; $\vec{wa}' = \vec{wa}$ except that $wa'(j) = wa(j) + wa(i)$ and $wa'(i) = 0$, and $\vec{ws}'$ "shifting" the components of $\vec{ws}$ according to the effect that relabeling $i \to j$ has on all possible weak shapes. We add $(wl, \vec{wa}', \vec{ws}')$ to U_r.

$r = \alpha_{i,j}(q)$**:** We initialize $U_r = U_q$. Then we repeat the following procedure as long as U_r is changing:

- Pick arbitrary $\vec{c} \in U_r$, $\vec{c} = (wl, \vec{wa}, \vec{ws})$ such that $wa(i) > 0$, and any weak shapes (a, B) and (b, C) such that $i \in B$, $b = j$, and $ws(a, B) > 0$, $ws(j, C) > 0$.
- Let $\vec{wa}' = \vec{wa}$ except that $wa'(i) = wa(i) - 1$.
- Let $\vec{ws}' = \vec{ws}$ except that $ws'(a, B) = ws(a, B) - 1$, $ws'(j, C) = ws(j, C) - 1$, and $ws'(a, B \cup C) = ws(a, B \cup C) + 1$.
- If $wa'(i) = 0$, then let the label i be subsequently "removed from" the label sets (of potentially active vertices) of all weak shapes indexing $\vec{ws}'$.
- Finally, add $(wl, \vec{wa}', \vec{ws}')$ to U_r.

IV. Output $\mathcal{U} = U_t$.

Proof. There are two steps in the proof.

Claim. The set $\mathcal{U}$ contains, for every annotated out-forest (F,μ) in G such that $\mu(x) \leq out_G(x) - out_F(x)$, the weak signature of (F,μ) for t.

This is easily proved by leaves-to-root structural induction on t: The claim is trivial at the leaves. Considering a node $r = p \oplus q$, the weak signature of any annotated out-forest in $G[r]$ that is obtained as a disjoint union of annotated out-forests in $G[p], G[q]$ of weak signatures $\vec{c}, \vec{d}$, respectively, equals computed $\vec{c} + \vec{d}$. Analogically for $r = \alpha_{i,j}(q)$. Notice that none of those two operations change potential activity of vertices by definition.

Consider one iteration at a node $r = \alpha_{i,j}(q)$. Let $(F + (u,v), \mu')$ be an annotated out-forest in $G[r]$ such that the weak signature $\vec{c}$ of (F,μ), $\mu(u) = \mu'(u) + 1$, has already been computed in previous iterations of U_r by the inductive assumption. Hence u of $G[r]$-label i is active in (F,μ) and $\vec{c}$ contains a weak shape (a,B) such that $i \in B$. Furthermore, since $F + (u,v)$ is an outforest, $\vec{c}$ contains a weak shape (b,C) such that $b = j$ is the $G[r]$-label of v. Then the vector $(wl, \vec{wa}', \vec{ws}')$ computed by the algorithm from $\vec{c}$ is exactly the weak signature of $(F + (u,v), \mu')$ by definition.

Claim. Algorithm 12 runs in XP time, i.e. in time $\mathcal{O}(n^{f(k)})$ where $f(k) \sim 2^{\mathcal{O}(k)}$.

The runtime of the algorithm is clearly dominated (up to a constant multiple of the exponent) by the number of possible weak signature vectors of length c. It is $c = 1+k+k2^k$. The value of each vector component may be a natural number up to n for $wl, \vec{ws}$ and up to n^2 for $\vec{wa}$. Hence the claim follows. ◀

The importance of Algorithm 12 comes from the following crucial statement.

▶ **Theorem 13.** *Suppose that the set $\mathcal{U} = U_t$ computed in Algorithm 12 contains a weak signature vector $\vec{w} = (wl, \vec{wa}, \vec{ws})$ such that $wl = \ell$, $\vec{wa} = \vec{0}$, and $\vec{ws}$ containing only one non-zero entry* 1 *(i.e. $\vec{w}$ corresponds to a weak signature of an out-tree with ℓ leaves and zero annotation). Then the graph $G = G[t]$ contains an out-branching with ℓ leaves.*

Consequently, Algorithm 12 solves the Minimum Leaf Out-Branching problem – for a given G and arbitrary ℓ – in XP-time wrt. the clique-width k of G (Theorem 1).

Proof. For a weak signature vector $\vec{c} \in U_s$ computed by Algorithm 12 on a subexpression s of the k-expression t, we introduce a *derivation tree* δ of $\vec{c}$ over t: δ is a rooted tree homeomorphic to that of s, each node of δ is labelled with one weak signature vector, and δ obtained from that of s as follows: First, each node r is replaced by a node labelled with some $\vec{c}_r \in U_r$, such that the root is labelled by $\vec{c}$ and for each edge $r \to q$ it holds that $\vec{c}_r$ is computed from $\vec{c}_q$ by the Algorithm 12. Second, we replace each edge $\vec{c}_r \to \vec{c}_q$, where $r = \alpha_{i,j}(q)$ and $\vec{c}_r$ was created from $\vec{c}_q$ by adding k edges, with a path $\vec{c}_r = \vec{c}_{r_0} \to \vec{c}_{r_1} \to \dots \vec{c}_{r_k} \to \vec{c}_q$ such that $\vec{c}_{r_l}$ is obtained from $\vec{c}_{r_{l+1}}$ by one (productive) iteration of the "$r = \alpha_{i,j}(q)$" step in III. Typically, one vector $\vec{c}$ can have many derivation trees.

Such a derivation tree δ is *realizable* over t if there exists an annotated out-forest (F,μ) in $G[s]$ such that, for each node d of δ, the corresponding subforest of (F,μ) has weak signature equal to the label of d. Obviously not all vectors in U_s have realizable derivations, in general.

Let $\mathcal{U}^o \subseteq \mathcal{U}$ be the set of *good vectors* assumed in the statement of this theorem, i.e. of those vectors $\vec{w} = (wl, \vec{wa}, \vec{ws}) \in \mathcal{U}$ such that $wl = \ell$, $\vec{wa} = \vec{0}$, and $\vec{ws}$ containing only one non-zero entry 1. Among all the good vectors $\vec{w} \in \mathcal{U}^o$, we select $\vec{w}_0$ and a derivation tree δ_0 of $\vec{w}_0$ such that there is a derivation tree $\delta_1 \subseteq \delta_0$ which is realizable over t and δ_1 maximizes the number of edges of its realizing out-forest (F_1, μ_1). We aim to show, by means of contradiction, that $\delta_1 = \delta_0$. Then (F_1, μ_1) would be a realization of whole δ_0 of weak signature $\vec{w}_0 \in \mathcal{U}^o$, and hence F_1 is an outbranching with ℓ leaves by the definition of $\mathcal{U}^o$.

Let $\delta_1 \subsetneq \delta_0$. Analyzing Algorithm 12.III, one easily finds out that both the "$r = p \oplus q$" and "$r = \rho_{i\to j}(q)$" operations preserve realizability. Hence we have got a realizing out-forest (F_1, μ_1) of δ_1 over t, its weak signature $\vec{c}_1$, and the label $\vec{c}_2$ of the parent of the root of δ_1 in the derivation tree δ_0 such that: $\vec{c}_2$ results from $\vec{c}_1$ by one iteration of the "$r = \alpha_{i,j}(q)$" rule, but no single edge of G can be added to (F_1, μ_1) to produce an out-forest of weak signature $\vec{c}_2$. In the rest of the proof we are going to construct another annotated out-forest with one more edge than (F_1, μ_1) such that its weak signature is contained in the derivation tree of some good vector in $\mathcal{U}^o$ (and this will be a contradiction to the assumptions).

We need a few more technical terms before proceeding with our proof.

- An *out-branching of a weak signature* vector $\vec{c}$ is any out-tree Γ such that $V(\Gamma)$ is the multiset of weak shapes respecting their multiplicities given by $\vec{c}$, i.e. every weak shape has the appropriate number of unique copies in $V(\Gamma)$. Informally, if $\vec{c}$ were realizable by an out-forest F, then the vertices of Γ would be all the out-trees of F.
- Considering a weak signature $\vec{c}$ labelling a node of a derivation tree δ, we say that an out-branching Γ of $\vec{c}$ is *determined by* δ if the following holds for every pair $x, y \in V(\Gamma)$: $(x, y) \in E(\Gamma)$ iff the computation run of Algorithm 12 associated with δ contains a "directed sequence" of $\alpha_{i,j}$ operations interconnecting the particular copies x to y.
- An out-branching Γ of a weak signature $\vec{c}$ is *feasible* for t if there exists good $\vec{d} \in \mathcal{U}^o$ such that a derivation tree δ of $\vec{d}$ contains the label $\vec{c}$ and Γ is determined by δ.

Informally, the out-branching Γ of $\vec{c}$ outlines the "intended arrangement" of components of $\vec{c}$ in a (potential) resulting out-branching of G.

In our case we have got an out-branching Γ_1 of the aforementioned weak signature $\vec{c}_1$ (of (F_1, μ_1)) determined by the derivation tree δ_0. Let $(x, y) \in E(\Gamma_1)$ be its edge such that x is a copy of the weak shape (a, B), $i \in B$ and y is a copy of the weak shape (j, C), and that $\vec{c}_2$ results from $\vec{c}_1$ in the iteration of the "$r = \alpha_{i,j}(q)$" rule (III) which picks the weak shapes (a, B) and (j, C) in $\vec{c}_1$. The digraph $\Gamma_1 - (x, y)$ has two weak components; X containing x and Y containing y. Let $F_1 = L_1 \cup L_1'$ be a partition of F_1 such that L_1 is formed by the out-trees corresponding to the vertices of X and L_1' is formed by those of Y, and, particularly, let $T_x \subseteq L_1, T_y \subseteq L_1'$ be the out-trees corresponding to x, y of Γ_1. Hence the root v_1 of T_y has F_1-label j and some potentially active vertex u_1 in T_x has F_1-label i,

We may as well assume that δ_1, its realization (F_1, μ_1) and x, y are chosen – subject to optimality in the previous criteria – such that they minimize the distance from the root of δ_1 to one of its nodes d_2 satisfying the following: In the annotated subforest (F_2, μ_2) induced from (F_1, μ_1) at the derivation node d_2 and containing u_1, there exists a vertex $u_2 \in V(F_2) \cap V(L_1)$ such that u_2 is active in (F_2, μ_2) and u_2 has the same F_2-label as u_1 (possibly $u_2 = u_1$). This leads to two cases to be considered:

i. The distance to our d_2 is zero. Then there is an active vertex $u_2 \in V(L_1)$ in (F_1, μ_1) of the F_1-label i, and so (u_2, v_1) is an edge of G.
ii. The distance to our d_2 is non-zero. Then, in particular, all active vertices of (F_1, μ_1) of the F_1-label i belong to L_1'.

Ad (i), we take the out-forest $F_1' = F_1 + (u_2, v_1) \subseteq G$. Let $\vec{c}_1{}'$ be the weak signature of the annotated out-forest (F_1', μ_1') where $\mu_1'(u_2) = \mu_1(u_2) - 1$ and $\mu_1'(x) = \mu_1(x)$ otherwise, and δ_1' the derivation tree of $\vec{c}_1{}'$ realizing (F_1', μ_1'). Let x' be the vertex of Γ_1 corresponding to the out-tree of F_1 containing u_2, let $\Gamma_1' = \Gamma_1 - (x, y) + (x', y)$, and Γ_1'' be obtained from Γ_1' by contracting (x', y). Clearly, Γ_1' is an out-branching of $\vec{c}_1$ and feasibility of Γ_1 naturally implies that also Γ_1' is feasible for t. Hence Γ_1'' is a feasible out-branching of $\vec{c}_1{}'$ for t. The derivation tree witnessing feasibility of Γ_1'' (in place of δ_0), its subtree δ_1' (in place of δ_1), and the annotated out-forest $(F_1' = F_1 + (u_2, v_1), \mu_1')$ contradict the optimality of our choice of $\vec{w}_0$ and δ_0 above. The proof is finished in this case (i).

Ad (ii), let the F_2-label of u_1 and u_2 be i'. Let d_3 be the parent node of d_2 in δ_1 and (F_3, μ_3) be induced from (F_1, μ_1) at d_3. By the optimality of our choice of d_2, the operation (cf. III) taking place at d_3 must be an iteration of $\alpha_{i',j'}$ adding an edge from u_2 (or u_2 would still be active in (F_3, μ_3)). Let $(u_2, v) \in E(F_3) \setminus E(F_2)$ be this added edge. Moreover, since u_1 is still potentially active in (F_1, μ_1), there exists a vertex $u_3 \in V(F_2) \cap V(L_1')$ of F_2-label i' active in (F_3, μ_3), and $(u_3, v) \in E(G)$.

Let $F_3' = F_2 + (u_3, v)$ and $\mu_3' = \mu_3$ except that $\mu_3'(u_2) = \mu_3(u_2) + 1$, $\mu_3'(u_3) = \mu_3(u_3) - 1$. The (F_3', μ_3') is an annotated out-forest in which u_2 is still active. Now, if u_3 is active in (F_1, μ_1), then we set $F_1' = F_1 - (u_2, v) + (u_3, v)$ and μ_1' accordingly. If u_3 is not active in (F_1, μ_1), then we pick any edge $(u_3, v') \in E(F_1) \setminus E(F_3)$ and subsequently define $F_1' = F_1 - (u_2, v) + (u_3, v) - (u_3, v') + (u_2, v')$ and $\mu_1' = \mu_1$. Again, it is routine to verify that F_1' is an out-forest in G in both cases. Let $\vec{c}_1{}'$ be the weak signature of new (F_1', μ_1') and δ_1' the derivation tree of $\vec{c}_1{}'$ realizing (F_1', μ_1').

Finally, we apply the computation run of the derivation tree δ_0 (starting up from the root of δ_1) onto the top of δ_1'. In this way we obtain an out-branching $\Gamma_1'$of $\vec{c}_1{}'$ that is an appropriate local modification of Γ_1. It follows from our choice of u_2, u_3 and their out-edges that Γ_1' is also feasible for t. Now we have an alternative optimal choice of $\vec{w}_0' \in \mathcal{U}^o$ and δ_0' which are witnessing feasibility of Γ_1', and of (F_1', μ_1') in place of (F_1, μ_1). This time, however, d_3 with δ_1' contradicts the optimality of our previous choice of d_2 (the distance from the root of δ_1' to d_3 is smaller by one).

This contradiction closes case (ii), and so the proof is finished. ◀

▶ **Question 14.** Seeing the complications in the proof of Theorem 13, one may naturally ask about a simpler solution of the problem. Say, cannot one come up with a better version of Definition 9 that, together with an appropriate modification of Lemma 10, would directly provide us with an XP algorithm? To be more formal, we ask whether there exists an equivalence relation $\sim$ on the set of annotated out-forests in a k-labelled graph H such that

i. $\sim$ refines $\approx_H$ (Definition 8) for every particular H, and
ii. the set of nonempty classes of $\sim$ for particular H can be computed dynamically over a k-expression of H in XP time.

4 Edge-disjoint paths on rank-width

▶ **Definition 15.** In the *Disjoint Paths problem*, an input is a graph (or digraph) G and k pairs of terminals $(s_1, t_1), \dots, (s_k, t_k)$, where $s_i, t_i \in V(G)$ for $1 \leq i \leq k$. The question is whether there exists a collection of k pairwise vertex-disjoint paths $P_1, \dots, P_k$ in G such that P_i connects s_i to t_i, $i = 1, \dots, k$.

The *Edge Disjoint Paths problem* is defined analogously with requiring the paths $P_1, \dots, P_k$ to be only pairwise edge-disjoint.

While the undirected Disjoint Paths variants are FPT solvable when parameterized simply by the number of paths (terminal pairs) [18], the directed case is NP-complete already for two paths in general. Hence it makes sense to look for suitable additional parameterizations of this problem, e.g. by clique-width. Note, on the other hand, that the Disjoint Paths problem with the number of paths k on the input is para-NP-complete for graphs of bounded clique-width [12], and the Edge Disjoint Paths problem with k on the input is para-NP-complete even for graphs of tree-width two [15].

▶ **Definition 16.** The *monadic second order logic* of one-sorted adjacency graphs, commonly abbreviated as MSO_1, has variables for graph vertices (say $x, y, z \dots$) and for vertex sets $(X, Y, Z \dots)$, common logic connectives and quantifiers, and a binary relational predicate *edge*. When dealing with directed graphs, we write *arc* instead of *edge*. Note that quantification over sets of edges is not possible (unlike in the more general MSO_2 language).

To give examples of MSO_1, we express that X is a dominating set in a graph G as $\delta(X) \equiv \forall y \notin X\, \exists z \in X\, edge(z, y)$, and that a digraph G is acyclic as $\alpha \equiv \forall X\, \exists y \in X\, \forall z \in X \neg arc(z, y)$. The MinLOB problem, on the other hand, is not expressible in MSO_1 (even with constant number of leaves) since neither the Hamiltonian Path is. Interestingly, the "dual" MaxLOB problem has an MSO_1 definition since, e.g. [10], a solution to MaxLOB is a complement to an out-connected dominating set in G.

Similarly, the (vertex) disjoint paths problem for a fixed k has a relatively easy description in MSO_1, e.g. [10]. For edge-disjoint paths with fixed k the situation is much more complicated – the inability to handle sets of edges seems to prevent us from expressing that two paths (possibly sharing many vertices) are indeed edge disjoint. Yet, surprisingly, with a few involved tricks we are able to express the existence of k directed pairwise edge-disjoint paths in MSO_1, and hence also to show membership in FPT when parameterized by clique-width or rank-width.

▶ **Theorem 17.** *Let G be a digraph, and $(s_1, t_1), \dots, (s_k, t_k)$ be pairs of terminals in G. There exists an MSO_1 formula π_k such that $G \models \pi_k(s_1, \dots, s_k, t_1, \dots, t_k)$ if, and only if, the corresponding* directed k edge-disjoint paths problem *in G has a solution.*

Proof. We start with an informal sketch of our approach. The initial idea is to "identify" each path P_i from s_i to t_i with its vertex set $X_i = V(P_i)$. Then, by standard means, we express the existence of a path from $s_i \in X_i$ to $t_i \in X_i$ as the nonexistence of a separation between s_i, t_i inside X_i. The difficult part is to express that two such paths P_i, P_j are edge-disjoint, while possibly sharing many vertices. The key to this lies in a deeper understanding of which edges are "really useful" for a potential (yet unknown) s_i–t_i-path induced on X_i. Imagine the much easier undirected case and X_i being inclusion-minimal – then $G[X_i]$ is actually an induced s_i–t_i-path and so our "really useful edges for P_i" could simply be identified as those edges having both ends in X_i. We now adapt this idea to the more difficult directed case. Let

$$\begin{aligned} \varrho(x, y, Z, r) \equiv\; & \forall Y \left[(x \in Y \wedge y \notin Y) \rightarrow \right. \\ & \left. \exists z, z' \in Z \left(z \in Y \wedge z' \notin Y \wedge z \neq r \neq z' \wedge \mathrm{arc}(z, z') \right) \right] \end{aligned} \tag{1}$$

be a formula stating that there exists a directed path from x to y on the vertices $Z \setminus \{r\}$ (note that $\{x, y\} \not\subseteq Z$ implies $G \not\models \varrho(x, y, Z, r)$), and put

$$\mu(s, t, Z, u, v) \equiv \varrho(s, u, Z, v) \wedge \varrho(v, t, Z, u) \tag{2}$$

Claim. Let Z be a vertex subset of G such that $s, t \in Z$ and $G[Z]$ contains a directed s–t-path. Then the following three statements hold:

(i) If $G \models \mu(s, t, Z, u, v)$, then $\{s, t, u, v\} \subseteq Z$.
(ii) If $G \models \neg\mu(s, t, Z, u, v)$, then no s–t-path in $G[Z]$ may contain the edge (u, v).
(iii) Suppose Z is inclusion-minimal such that $G[Z]$ contains a s–t-path P. Then such P is unique and $E(P)$ is the set of those $(u, v) \in E(G)$ such that $G \models \mu(s, t, Z, u, v)$.

For (i) the proof follows directly from the definition of $\mu(s,t,Z,u,v)$. To see that (ii) also holds, it is enough to note that every edge (u,v) of every s–t-path in $G[Z]$ satisfies $G \models \mu(s,t,Z,u,v)$ by (1). Finally to prove (iii) let us suppose that $(u,v) \in E(G[Z]) \setminus E(P)$ (i.e., (u,v) points "backwards" on P due to minimality of Z). If in (2), for instance, $G \models \varrho(s,u,Z,v)$, then the corresponding s–u-path joined with the u–t-subpath of P would result in an s–t-path in $G[Z]$ avoiding v, a contradiction to minimality of Z. This finishes the proof of the claim.

Now, (iii) provides us with a criterion for identifying edges used by one particular s–t-path. To make use of it in a k path problem, we have to identify edges used by the first path P_1 in G, then edges used by P_2 in $G - E(P_1)$, then those used by P_3 in $G - E(P_1 \cup P_2)$, etc. For that we use the following trick which "replaces" the atomic predicate *arc* in (1) with appropriate recursively defined (3) formulas α_j where $j = 1,\dots,k$. For simplicity, we write $\widehat{s_j}$ as a shortcut for the list $s_1, s_2 \dots, s_j$, and analogically for $\widehat{t_j}, \widehat{X_j}$.

$$\begin{aligned} \alpha_1(u,v) &\equiv \operatorname{arc}(u,v)\,, \\ \alpha_{j+1}(u,v,\widehat{s_j},\widehat{t_j},\widehat{X_j}) &\equiv \alpha_j(u,v,\widehat{s_{j-1}},\widehat{t_{j-1}},\widehat{X_{j-1}}) \wedge \neg\mu_j(s_j,t_j,X_j,u,v,\widehat{s_{j-1}},\widehat{t_{j-1}},\widehat{X_{j-1}}) \end{aligned} \tag{3}$$

where $\mu_j \equiv \varrho_j(s_j,u,X_j,v,\widehat{s_{j-1}},\widehat{t_{j-1}},\widehat{X_{j-1}}) \wedge \varrho_j(v,t_j,X_j,u,\widehat{s_{j-1}},\widehat{t_{j-1}},\widehat{X_{j-1}})$ analogically to (2), and ϱ_j is replacing the *arc* predicate in ϱ (1) simply as follows

$$\begin{aligned} \varrho_j(x,y,Z,r,\widehat{s_{j-1}},\widehat{t_{j-1}},\widehat{X_{j-1}}) \;\equiv\; & \forall Y \left[(x \in Y \wedge y \notin Y) \rightarrow \right. \\ & \left. \exists z,z' \in Z \left(z \in Y \wedge z' \notin Y \wedge z \neq r \neq z' \wedge \alpha_j(z,z',\widehat{s_{j-1}},\widehat{t_{j-1}},\widehat{X_{j-1}})\right)\right]. \end{aligned} \tag{4}$$

Writing just ϱ'_j in place of previous ϱ_j "without r" (4), we obtain the solution

$$\begin{aligned} \pi_k(\widehat{s_k},\widehat{t_k}) \;\equiv\; & \exists \widehat{X_k}\; \varrho'_1(s_1,t_1,X_1) \wedge \varrho'_2(s_2,t_2,X_2,\widehat{s_1},\widehat{t_1},\widehat{X_1}) \wedge \\ & \cdots \wedge \varrho'_k(s_k,t_k,X_k,\widehat{s_{k-1}},\widehat{t_{k-1}},\widehat{X_{k-1}})\,. \end{aligned}$$

It remains to prove that $G \models \pi_k(\widehat{s_k},\widehat{t_k})$ if, and only if, there exist k edge-disjoint s_i–t_i paths in G where $i = 1,\dots,k$. In one direction, suppose a particular choice of the vertex sets $\widehat{X_k}$ satisfying π_k on G. According to (3) and (4) this assumption means that, for each $i = 1,\dots,k$ by induction, there exists a directed s_i–t_i path P_i on the vertices X_i such that P_i completely avoids (ii) edges potentially usable by the paths $P_1,\dots,P_{i-1}$. Hence such $P_1,\dots,P_k$ are pairwise edge-disjoint in G.

Conversely, among all collections of k pairwise edge-disjoint s_i–t_i paths P_i in G, we select one lexicographically minimizing the vector $\big(len(P_1),\dots,len(P_k)\big)$. Then, clearly, each $X_i = V(P_i)$ for $i = 1,\dots,k$ is inclusion-minimal inducing an s_i–t_i path in $G - E(P_1 \cup \dots \cup P_{i-1})$, and so the claim (iii) applies here. Hence, by induction on i, we conclude from (3) that $G \models \alpha_{i+1}(u,v,\widehat{s_i},\widehat{t_i},\widehat{X_i})$ iff $(u,v) \in E(G) \setminus E(P_1 \cup \dots \cup P_i)$. And since $P_{i+1} \subseteq G - E(P_1 \cup \dots \cup P_i)$, it follows from (4) that our selected sets $X_1,\dots,X_k$ satisfy $\pi_k(\widehat{s_k},\widehat{t_k})$ on G. ◀

In connection with [3][1] we finally obtain:

▶ **Corollary 18.** *Both the undirected and directed edge-disjoint paths problems with fixed k have a linear FPT algorithm on simple (di)graphs of bounded clique-width.*

[1] Note that [3] consider only undirected graphs, but the same results also hold for digraphs, cf. [14, 8].

▶ **Question 19.** Notice that the MSO_1 formula π_k constructed in the proof of Theorem 17 has quantifier alternation depth growing with k. Therefore the worst-case runtime estimate of Corollary 18 coming from [3] has a tower-exponential dependency on the parameter k. The question thus is whether an MSO_1 description of the k edge-disjoint paths problem is possible with fixed quantifier alternation depth. (An ad-hoc estimate of the Myhill–Nerode congruence classes of the problem suggests this might be true.)

References

1 J. Bang-Jensen and G. Gutin. *Digraphs: Theory, Algorithms and Applications.* Springer Monographs in Mathematics. Springer, second edition, 2009.

2 D. Corneil and U. Rotics. On the relationship between cliquewidth and treewidth. *SIAM J. Comput.*, 34(4):825–847, 2005.

3 B. Courcelle, J. A. Makowsky, and U. Rotics. Linear time solvable optimization problems on graphs of bounded clique-width. *Theory Comput. Syst.*, 33(2):125–150, 2000.

4 B. Courcelle and S. Olariu. Upper bounds to the clique width of graphs. *Discrete Appl. Math.*, 101(1-3):77–114, 2000.

5 P. Dankelmann, G. Gutin, and E. Kim. On complexity of minimum leaf out-branching problem. *Discrete Appl. Math.*, 157(13):3000–3004, 2009.

6 W. Espelage, F. Gurski, and E. Wanke. How to solve NP-hard graph problems on clique-width bounded graphs in polynomial time. In *WG'01*, volume 2204 of *LNCS*, pages 117–128. Springer, 2001.

7 F. Fomin, P. Golovach, D. Lokshtanov, and S. Saurab. Clique-width: On the price of generality. In *SODA'09*, pages 825–834. SIAM, 2009.

8 R. Ganian and P. Hliněný. On parse trees and Myhill–Nerode–type tools for handling graphs of bounded rank-width. *Discrete Appl. Math.*, 158:851–867, 2010.

9 R. Ganian, P. Hliněný, J. Kneis, A. Langer, J. Obdržálek, and P. Rossmanith. On digraph width measures in parametrized algorithmics. In *IWPEC'09*, volume 5917 of *LNCS*, pages 161–172. Springer, 2009.

10 R. Ganian, P. Hliněný, J. Kneis, A. Langer, J. Obdržálek, and P. Rossmanith. Digraph width measures in parametrized algorithmics. Submitted. Available from: `http://www.fi.muni.cz/~hlineny/Research/papers/kenny-full.pdf`, 2010.

11 R. Ganian, P. Hliněný, and J. Obdržálek. Unified approach to polynomial algorithms on graphs of bounded (bi-)rank-width. Submitted, 2010. 29 p.

12 F. Gurski and E. Wanke. Vertex disjoint paths on clique-width bounded graphs. *Theor. Comput. Sci.*, 359(1-3):188–199, 2006.

13 P. Hliněný and S. Oum. Finding branch-decomposition and rank-decomposition. *SIAM J. Comput.*, 38:1012–1032, 2008.

14 M. Kanté. The rank-width of directed graphs. arXiv:0709.1433v3, March 2008.

15 T. Nishizeki, J. Vygen, and X. Zhou. The edge-disjoint path problem is NP-complete for series-parallel graphs. *Discrete Appl. Math.*, 115(1-3):177–186, 2001.

16 S. Oum and P. D. Seymour. Approximating clique-width and branch-width. *J. Combin. Theory Ser. B*, 96(4):514–528, 2006.

17 N. Robertson and P. D. Seymour. Graph minors. II. Algorithmic aspects of tree-width. *J. Algorithms*, 7(3):309–322, 1986.

18 N. Robertson and P. D. Seymour. Graph minors. XIII: The disjoint paths problem. *J. Comb. Theory Ser. B*, 63(1):65–110, 1995.

19 G. Salamon and G. Wiener. On finding spanning trees with few leaves. *Inform. Process. Lett.*, 105(5):164–169, 2008.

Appendix

▶ **Lemma 10.** *Let* (F_1, μ_1) *and* (F_2, μ_2) *be a pair of annotated out-forests in a* k*-labelled digraph* H*. If the signatures of* (F_1, μ_1) *and* (F_2, μ_2) *are equal, then* $(F_1, \mu_1) \approx_H (F_2, \mu_2)$*.*

Proof. For the proof of Lemma 10, it is useful to define also the *strong signature* of (F, μ) which, in addition to ordinary signature, counts the out-trees $T_0 \subseteq F$ according to their full shape. The *full shape* of T_0 is the pair $fs(T_0) = (a, \vec{m})$ where a is the H-label of the root of T_0, and $m(i)$ equals the sum of $\mu(x)$ over all vertices $x \in V(T_0)$ of the label i for $i = 1, \ldots, k$. Trivially, if the strong signatures of (F_1, μ_1) and (F_2, μ_2) are equal, then $(F_1, \mu_1) \approx_H (F_2, \mu_2)$. The size of a strong signature vector, unfortunately, grows with H.

Assume, for a contradiction, that $(F_1, \mu_1) \not\approx_H (F_2, \mu_2)$ and that the Hamming distance of the strong signature vectors of (F_1, μ_1) and of (F_2, μ_2) is minimized (only manipulation with μ's is needed here). Namely, this minimum distance is two and there exist out-trees $T_1, T_1' \subseteq F_1$ and $T_2, T_2' \subseteq F_2$ such that the only difference between the strong signatures of (F_1, μ_1) and (F_2, μ_2) is in $fs(T_1) \neq fs(T_2)$, $fs(T_1') \neq fs(T_2')$ while the (ordinary) shapes of T_1, T_2 and of T_1', T_2' equal. Let $fs(T_1) = (a, \vec{m}_1)$, $fs(T_2) = (a, \vec{m}_2)$ and $fs(T_1') = (a', \vec{m}_1')$, $fs(T_2') = (a', \vec{m}_2')$. By further minimization we may assume that the only difference between $fs(T_1), fs(T_2)$ and between $fs(T_1'), fs(T_2')$ is due to $m_1(i) = m_2(i) + 1$ and $m_2'(i) = m_1'(i) + 1$. Let $v_1 \in V(T_1)$ and $v_2' \in V(T_2')$ be the two (active) vertices accounting for this difference. Notice that since the shapes of T_1, T_2 equal, $m_2(i) > 0$ and so $m_1(i) \geq 2$.

Since we assume $(F_1, \mu_1) \not\approx_H (F_2, \mu_2)$ (Definition 8), up to symmetry, there is $G \supseteq H$ such that (F_1, μ_1) can be extended to an out-branching $U_1 \subseteq G$ with $\leq \ell$ leaves while (F_2, μ_2) cannot be. Let $e_1, f_1 \in E(U_1) \setminus E(H)$ be two distinct edges such that e_1 starts (has tail) in v_1 and f_1 starts in any vertex of T_1 of label i. Such edges exist since $\mu_1(v_1) > 0$ and $m_1(i) \geq 2$. Let $A_1 \subseteq E(U_1) \setminus E(H)$ be the set of edges incident with the vertices of H, excluding the edge e_1. By our previous assumption restricting the difference between the strong signatures of (F_1, μ_1) and (F_2, μ_2), and by the fact that H is a labelled-modular subdigraph of G, there clearly exists a natural bijection $b : A_1 \to A_2 \subseteq E(G) \setminus E(H)$ satisfying the following: The spanning subgraph $U_2 \subseteq G$ consisting of F_2, the edges of A_2, and the edges of $E(U_1) \setminus (A_1 \cup E(H))$, is an out-forest with two weak components.

Let $T_3 \subseteq U_2$ be the out-tree whose root v_3 is the head of e_1 (if the head of e_1 was in H – a degenerate case, then the root r of some component of U_2 has the same H-label as the head of e_1, and we choose $v_3 = r$ instead). If $v_2' \notin V(T_3)$, then we simply add the edge (v_2', v_3) to U_2 and obtain an out-branching of G extending (F_2, μ_2) with the same number of leaves as U_1. If $v_2' \in V(T_3)$, then we take the out-forest $U_2 - b(f_1)$ and add the two edges (u_3, v_3), (v_2', u_4) where $(u_3, u_4) = b(f_1)$, again obtaining an out-branching of G. These edges must exist in G since H is a labelled-modular subdigraph of G. Both the conclusions are contradicting $(F_1, \mu_1) \not\approx_H (F_2, \mu_2)$ and so the theorem is proved. ◀