BOOLEAN DIMENSION AND TREE-WIDTH

STEFAN FELSNER, TAMÁS MÉSZÁROS, AND PIOTR MICEK

Abstract. Dimension is a key measure of complexity of partially ordered sets. Small dimension allows succinct encoding. Indeed if P has dimension d, then to know whether $x \leqslant y$ in P it is enough to check whether $x \leqslant y$ in each of the d linear extensions of a witnessing realizer. Focusing on the encoding aspect, Nešetřil and Pudlák defined a more expressive version of dimension. A poset P has boolean dimension at most d if it is possible to decide whether $x \leqslant y$ in P by looking at the relative position of x and y in only d linear orders on the elements of P (not necessarilly linear extensions). We prove that posets with cover graphs of bounded tree-width have bounded boolean dimension. This stands in contrast with the fact that there are posets with cover graphs of tree-width three and arbitrarily large dimension. This result might be a step towards a resolution of the long-standing open problem: Do planar posets have bounded boolean dimension?

1. Introduction

Partially ordered sets, called *posets* for short, are combinatorial structures with applications in various mathematical fields, e.g. set theory, topology, algebra, and theoretical computer science. The most important measure of a poset's complexity is its dimension.

A *linear extension* L of a poset P is a total order on the elements of P such that if $x \leqslant y$ in P then $x \leqslant y$ in L. A *realizer* of a poset P is a set $\{L_1, \ldots, L_d\}$ of linear extensions of P such that

$$x \leqslant y \Longleftrightarrow (x \leqslant y \text{ in } L_1) \wedge \cdots \wedge (x \leqslant y \text{ in } L_d),$$

for every $x, y \in P$. The *dimension* of P, denoted by $\dim(P)$, is the minimum size of its realizer.

The realizers provide a way of succinctly encoding posets. Indeed, if a poset is given with a realizer witnessing dimension d, then a query of the form "is $x \leqslant y$?" can be answered by looking at the relative position of x and y in each of the d linear extensions of the realizer. This application motivates the following presumably more efficient encoding of posets proposed by Nešetřil and Pudlák [12], following the work of Gambosi, Nešetřil and Talamo [8].

(S. Felsner) Institut für Mathematik, Technische Universität Berlin
(T. Mészáros) Institut für Mathematik, Freie Universität Berlin
(P. Micek) Faculty of Mathematics and Computer Science, Jagiellonian University
E-mail addresses: felsner@math.tu-berlin.de, tamas.meszaros@fu-berlin.de, piotr.micek@tcs.uj.edu.pl.

Key words and phrases. posets, tree-width, boolean dimension.
Piotr Micek was partially supported by the National Science Center of Poland, grant no. 2015/18/E/ST6/00299.
Tamás Mészáros was supported by the Dahlem Research School of Freie Universität Berlin.

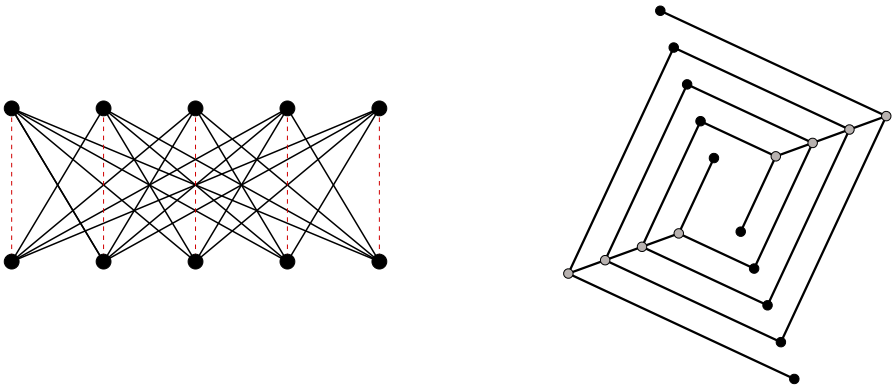

FIGURE 1. The standard example S_5 (left). Kelly's planar poset containing an induced S_5 (right).

A *boolean realizer* of a poset P is a set of linear orders (not necessarily linear extensions) $\{L_1, \ldots, L_d\}$ of elements of P for which there exists a d-ary boolean formula ϕ such that

$$x \leqslant y \iff \phi((x \leqslant y \text{ in } L_1), \ldots, (x \leqslant y \text{ in } L_d)) = 1,$$

for every $x, y \in P$. The *boolean dimension* of P, denoted by $\mathrm{bdim}(P)$, is a minimum size of a boolean realizer. Clearly, for every poset P we have

$$\mathrm{bdim}(P) \leqslant \dim(P).$$

The usual dimension of a poset on n elements may be linear in n. The so-called, *standard example* S_n, for $n \geqslant 2$, is a poset on $2n$ elements $a_1, \ldots, a_n, b_1, \ldots, b_n$ with $a_i < b_j$ in S_n if and only if $i \neq j$, and with no other comparabilities (see Figure 1). On the other hand, in their seminal paper Dushnik and Miller [4] observed that $\dim(S_n) = n$. It is a nice little exercise to show that $\mathrm{bdim}(S_n) \leqslant 4$ for every n. In general, Nešetřil and Pudlák [12] showed that boolean dimension of posets on n elements is $O(\log n)$. They also provide an easy counting argument showing that there are posets on n elements with boolean dimension at least $c \log n$, where c is some positive constant.

The *cover graph* of a poset P is the graph on the elements of P with edge set $\{xy \mid x < y \text{ in } P \text{ and there is no } z \text{ with } x < z < y \text{ in } P\}$. A poset is *planar* if it has a planar diagram, i.e., its cover graph has a non-crossing upward drawing in the plane. This means that every edge xy with $x < y$ is drawn as a curve that goes monotonically up from the point of x to the point of y. Somewhat unexpectedly, planar posets have arbitrarily large dimension. Kelly [9] gave a construction that embeds a standard example as a subposet into a planar poset, see Figure 1. Another property of Kelly's construction is that the cover graphs of resulting posets have tree-width (and even path-width) at most 3. Still, the boolean dimension of standard examples and Kelly's construction is at most 4. There is a beautiful open problem posed in [12] that remains a challenge with essentially no progress over the years.

Problem 1 (Nešetřil and Pudlák (1989))**.** *Is the boolean dimension of planar posets bounded?*

Nešetřil and Pudlák suggested an approach for the negative resolution of this problem that involves an auxiliary Ramsey-type problem for planar posets. From the positive side, Brightwell and Franciosa [3] in 1996 proved that spherical posets with a least element

have bounded boolean dimension, contrary to ordinary dimension. More recently, there was a considerable effort to understand the general problem whether posets with stucturally restricted cover graphs have bounded boolean dimension. Mészáros, Micek and Trotter [10] proved that boolean dimension of a poset is bounded in terms of boolean dimension of its 2-connected blocks. Micek and Walczak [11] proved that posets with cover graphs of bounded path-width have bounded boolean dimension.

The contribution of our paper is the following result.

Theorem 2. *Posets with cover graphs of bounded tree-width have bounded boolean dimension.*

The usual dimension is known to be at most 3 for posets with cover graphs being forests (Trotter, Moore [15]) and at most 12 for posets with cover graphs of tree-width 2 (Seweryn [13]). As mentioned before, Kelly's examples have tree-width 3 and arbitrarily large dimension. This certifies that boolean realizers are able to represent natural classes of posets that are out of reach in the default setting.

It is tempting to ask whether a result similar to Theorem 2 holds for broader classes of sparse posets. Besides planar posets, it might be true even for posets whose cover graphs exclude a fixed graph as a minor. However, we note that excluding a fixed graph as a topological minor is insufficient. Indeed, there are posets whose cover graphs have maximum degree at most 3 and that have unbounded boolean dimension. For completeness we include such an example in Section 6.

In 2016, Ueckerdt [16] proposed yet another variant of poset's dimension, the *local dimension*. Interestingly, local dimension of posets with bounded path-width is bounded, while it becomes unbounded for posets of tree-width 3 (Barrera-Cruz et al. [1]). It is also known that planar posets have unbounded local dimension (Bosek, Grytczuk and Trotter [2]).

This paper is organized as follows. In Section 2, we proceed with the necessary definitions. In particular, we introduce branching programs which we will use to build our formulas in the boolean realizers. We present two simple but important subroutines that we use later on extensively. In Section 3, we set up the proof of Theorem 2 and build some auxiliary structures and colorings based on the poset given on the input and on the tree-decomposition of its cover graph. In Section 4, we prove Theorem 2, while in Section 5, we present a connection of boolean dimension to labeling schemes for reachability queries. In particular, we discuss how a positive resolution of Problem 1 would imply the existence of a labeling scheme of size $O(\log n)$ for reachability queries for planar digraphs. Finally, in Section 6, we discuss an example of posets whose cover graphs have maximum degree at most 3 and that have unbounded boolean dimension

2. Preliminaries

2.1. **Tree-decompositions.** Let G be a graph. A *tree-decomposition* of G is a pair $(T, \{B_t\}_{t\in V(T)})$ where T is a tree and $\{B_t\}_{t\in V(T)}$ is a family of subsets of $V(G)$ satisfying:

(i) for each $v \in V(G)$ there exists $t \in V(T)$ with $v \in B_t$;
(ii) for every edge $uv \in E(G)$ there exists $t \in V(T)$ with $u, v \in B_t$;
(iii) for each $v \in V(G)$, if $v \in B_t \cap B_{t''}$ for some $t, t'' \in V(T)$, and t' lies on the path in T between t and t'', then $v \in B_{t'}$.

By Property (iii), we have that for every vertex $v \in V(G)$, the vertices $t \in V(T)$ for which $v \in B_t$ form a subtree of T, called the *subtree of* v.

The quality of a tree-decomposition $(T, \{B_t\}_{t \in V(T)})$ is usually measured by its width, i.e. the maximum of $|B_t| - 1$ over all $t \in V(T)$. Then the *tree-width* $\mathrm{tw}(G)$ of G is the minimum width of a tree-decomposition of G.

2.2. **Branching programs.** To show that a poset P has $\mathrm{bdim}(P) \leqslant d$, one should provide d linear orders $\pi_1, \ldots, \pi_d$ of the elements of P and a boolean formula $\phi(\xi_1, \ldots, \xi_d)$. It will be convenient to describe the formula ϕ as a branching program.

We think of a branching program $\mathcal{B}$ as a rooted tree. The nodes of the tree are subprograms. The task of a subprogram at node N is to return a boolean value to its parent, this boolean value is called the *evaluation* of N. The evaluation of N is a function of the values returned from the children and of the input variables $\xi_1, \ldots, \xi_d$ of $\mathcal{B}$. The evaluation $\mathrm{eval}(\mathcal{B})$ of the branching program $\mathcal{B}(b_1, \ldots, b_d)$ with inputs $b_1, \ldots, b_d$ is the evaluation of its root node. A branching program $\mathcal{B}$ is said to represent a boolean formula $\phi(\xi_1, \ldots, \xi_d)$ if for all inputs $b_1, \ldots, b_d$, we have $\mathrm{eval}(\mathcal{B}) = \phi(b_1, \ldots, b_d)$.

Therefore we can prove $\mathrm{bdim}(P) \leqslant d$ by describing a branching program that answers queries '$(x < y)$?' using variables $\xi_1, \ldots, \xi_d$, where $\xi_i = 1$ if and only if $x < y$ in π_i. In this case, we say that a branching program *depends* on $\pi_1, \ldots, \pi_d$.

2.3. **Tools.** The first tool is a straightforward branching program able to detect if a queried element lies within some fixed set. For a linear order π on a set V let π^* denote the reversal of π and for $X \subseteq V$, let $\pi(X)$ denote the projection of π onto X. We also work with linear orders as sequences, i.e., when A is disjoint from B, π is a linear order on A and σ is a linear order on B, then $\pi\sigma$ is a concatenation of these linear orders which is a linear order on $A \cup B$.

Tool 1 (Set Membership)**.** *Let V be a set and let $C \subseteq V$. Then there are three linear orders on V such that for every distinct $x, y \in V$ by looking at the order of x and y in these three linear orders one can decide whether each x and y belong to C or not.*

Proof. Fix an arbitrary linear order π on V and consider the following three linear orders: $\pi_1 = \pi(C)\pi(V - C)$, $\pi_2 = \pi^*(C)\pi(V - C)$ and $\pi_3 = \pi(C)\pi^*(V - C)$ do the job. Then $x \in C$ if and only if x and y switch orders in π_1 and π_2 or in all three linear orders $x < y$. Similarly, $y \in C$ if and only if x and y switch orders in π_1 and π_2 or in all three linear orders $y < x$. □

The other tool deals with elements lying in a tree. Let T be a rooted tree with some of its edges being colored either with Red or with Green. For a vertex $u \in V(T)$, we denote the subtree of T rooted at u by T_u. If $v \in T_u$, then we say that u is *below* v in T or v is *above* u in T. Note that, in particular, u is below itself. For $u, v \in V(T)$ we denote the unique path between u and v in T by $[u, v]$, and the vertex in $[u, v]$ that is closest to the root of T by $u \wedge v$. We call $u \wedge v$ the *meet* of u and v. We also assume that T is given together with a planar upward drawing with lowest vertex being the root. This implies that at every vertex u there is a left-to-right ordering of the subtrees rooted at the children of u. Now suppose that the vertices u and v are such that none of them is below the other. Then $u \wedge v$ has two children $u' \neq v'$ such that $u \in T_{u'}$ and $v \in T_{v'}$. We say that u *is left of* v in T if $T_{u'}$ comes before $T_{v'}$ in the left-to-right ordering of the

subtrees rooted at the children of $u \wedge v$. For example, in Figure 2.3 we have that c is below r, the meet of f and s is a, and k is left of n.

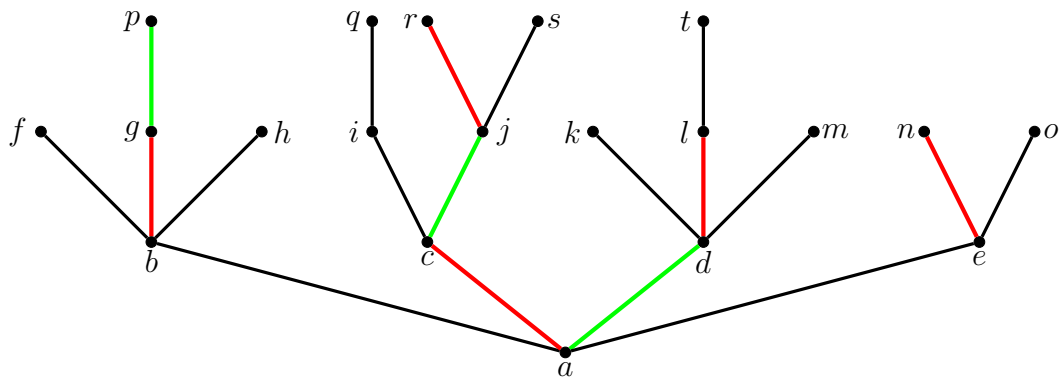

FIGURE 2. A RED - GREEN-colored tree. With this tree at the input, Algorithm 1, when using the topological order that is represented by the alphabetical labeling of the vertices, returns the linear order

$$c\ i\ q\ r\ j\ s\ g\ p\ n\ a\ b\ e\ f\ h\ o\ l\ t\ d\ k\ m,$$

while Algorithm 2 returns the linear order

$$a\ b\ f\ h\ d\ k\ m\ l\ t\ e\ o\ n\ c\ i\ q\ j\ s\ r\ g\ p.$$

Tool 2 (Color Detection)**.** *Let T be a rooted tree with some of its edges being colored either with* RED *or with* GREEN*. Then there is a branching program $\mathcal{B}$ depending on five linear orders on $V(T)$ such that for every queried $x, y \in V(T)$ with $x \wedge y$ being strictly below x (resp. y) in T, $\mathcal{B}$ outputs 1 if and only if the first colored edge on $[x \wedge y, x]$ (resp. $[x \wedge y, y]$) is* RED*.*

Proof. The two setups are clearly symmetric, so in the proof, we concentrate only on the case when the queried vertices x, y are such that $x \wedge y$ is strictly below x.

The root node N_{root} of $\mathcal{B}$ establishes first the relative position of x and y in T. This can be done with two linearorders on $V(T)$: the left-to-right depth first search ordering π_L of $V(T)$ and the right-to-left depth-first-search ordering π_R of $V(T)$. By the assumption $x \wedge y$ being strictly below x in T, there are three possible outcomes:

- y is strictly below x in $T \iff y < x$ in both π_L and π_R;
- x is left of y in $T \iff x < y$ in π_L and $y < x$ in π_R;
- y is left of x in $T \iff y < x$ in π_L and $x < y$ in π_R.

N_{root} has three children $N_{y \text{ below } x}$, $N_{x \text{ left of } y}$ and $N_{y \text{ left of } x}$, and it simply outputs the evaluation of the child responsible for the detected case of the relative position of x and y in T. We will show that each child needs only one extra linear order to output the correct value. We start with $N_{y \text{ below } x}$.

Claim. *Suppose y is strictly below x and let π be the linear order on $V(T)$ produced by Algorithm 1. Then $x < y$ in π if and only if the first colored edge on the path $[y, x]$ in T is* RED*.*

Algorithm 1: Color detection for y being below x in T

```
Procedure process(v)
    C(v) = {u ∈ T_v | the path [v, u] has no colored edges}
    R(v) = {u ∈ T_v | the path [v, u] has a unique colored edge,
                      this edge is the last edge and it is RED}
    G(v) = {u ∈ T_v | the path [v, u] has a unique colored edge,
                      this edge is the last edge and it is GREEN}
    for u ∈ R(v) do
        process(u)
    L = list of all vertices in C(v) in a topological order in T
    append L to π
    for u ∈ G(v) do
        process(u)
π = empty
process(root(T))
return π
```

Proof of the Claim. We start with two simple observations.

- During the execution of process(root(T)), every vertex of T is a member of $C(v)$ for a unique $v \in V(T)$.
- A call of process(v) appends all elements of T_v to π before returning.

Now consider x, y such that y is strictly below x in T, and let z_y be the node of T for which $y \in C(z_y)$ during the algorithm. Also let L_y be the list of all vertices in $C(z_y)$ in topological order. We distinguish three cases.

(i) *The path $[y, x]$ has no colored edges.* In this case $x \in C(z_y)$, and in L_y we have y before x. Therefore $y < x$ in π.

(ii) *The first colored edge (v, w) on the path $[y, x]$ is* GREEN. In this case $w \in G(z_y)$, hence the process(z_y) appends the list L_y containing y before calling process(w). However, as process(w) is the one which puts x in the linear order, we again have $y < x$ in π.

(iii) *The first colored edge (v, w) on the path $[y, x]$ is* RED. In this case $w \in R(z_y)$ and hence the process(z_y) appends call process(w), and puts x in π, before appending the list L_y containing y. Now we have $x < y$ in π.

□

We move on to $N_{x \text{ left of } y}$.

Claim. *Suppose x is left of y and let π be the linear order on $V(T)$ produced by Algorithm 2. Then $y < x$ in π if and only if the first colored edge on the path $[x \wedge y, x]$ in T is* RED.

Proof of the Claim. We start the proof again with some simple observations.

- For each vertex v in T, there is a unique call of process(v), and so the resulting π is really a linear order on $V(T)$.

Algorithm 2: Color detection for x being left of y in T

```
Procedure process(v)
    append v to π
    for each w child of v in T taken in left-to-right-order do
        if (v, w) is uncolored then
            process(w)
        if (v, w) is RED then
            S.push(w)
        if (v, w) is GREEN then
            S.push(v)                // marking the beginning of the local stack of v
            process(w)
            while S.top() ≠ v do
                process(S.pop())
            S.pop()                  // taking off the marker
π = ∅
S = empty stack
process(root(T))
while S is non-empty do
    process(S.pop())
return π
```

- If u, v, u', v' are vertices of T such that $u \in T_{u'}$ and $v \in T_{v'}$ and u' is on the current stack when process(v') is called, then $v < u$ in π.
- If $[u, u']$ is a path of uncolored edges in T and v is a vertex that is above u and right of u', then $u' < v$ in π.
- If (v, w) is a GREEN edge with v below w, then the local stack of v makes the procedure behave as if Algorithm 2 had been called for the tree T_w. In particular it appends all the vertices of T_w in a consecutive block of π.

Now consider x, y such that x is left of y in T. We again distinguish three cases.

(i) *The path* $[x \wedge y, x]$ *has no colored edges.* Then $x < y$ in π follows from the third observation above.

(ii) *The first colored edge* (v, w) *on the path* $[x \wedge y, x]$ *is* GREEN. From the third observation above, we obtain $v < y$ in π. By the fourth observation, the call of process(w) appends all vertices of T_w, including x, to π in a consecutive block before the processing of the local stack of v is finished, and so before y is touched. This implies that $x < y$ in π.

(iii) *The first colored edge* (v, w) *on the path* $[x \wedge y, x]$ *is* RED. Then w is put on the stack when processing v and remains on the stack until all children of $x \wedge y$ have been processed. If v' is the child of $x \wedge y$ with $y \in T_{v'}$, then w is on the stack when process(v') is called. The second observation above shows that in this case $y < x$ in π.

□

As the cases 'x left of y' and 'y left of x' are clearly symmetric, this finishes the proof of the existence of Tool 2. □

3. The proof setup

Let P be a poset with cover graph G and suppose that $\mathrm{tw}(G) \leqslant k$. Fix a tree-decomposition $(T, \{B_t\}_{t \in V(T)})$ of G of width at most k. We imagine the tree T to be rooted and being drawn upwards with lowest vertex the root. In particular at every vertex t there is a fixed left-to-right order of its children. For $z \in P$ let $\mathrm{root}(z)$ denote the root (i.e. the lowest vertex) of the subtree of z. Massaging a bit the tree-decomposition, we can assume that the vertices $\mathrm{root}(z)$, $z \in P$ are all distinct.

First, we apply a standard greedy coloring procedure to the elements of P: Fix any ordering of the elements of P such that if $\mathrm{root}(z)$ is below $\mathrm{root}(z')$ in T, then z is before z' in the ordering. Then, along this ordering, color an element $z \in P$ with the least possible color that does not appear in $B_{\mathrm{root}(z)}$. In this way, the elements from the same bag will have distinct colors, and we clearly use at most $k+1$ colors. Denote this coloring by $c : P \to [k+1]$. For a vertex $t \in V(T)$ and $i \in [k+1]$, if there is a (unique) element $z \in B_t$ of color i, then we call it the *representative* of color i at t and denote it by $\mathrm{repr}_i(t)$, otherwise we say that $\mathrm{repr}_i(t)$ is undefined.

For a vertex $t \in V(T)$ and an element $z \in P$ such that $\mathrm{root}(z)$ is below t in T, we define the vector $\mathrm{vec}(z,t)$ of length $k+1$ so that for $i \in [k+1]$, the ith coordinate is

$$\mathrm{vec}_i(z,t) = \begin{cases} < & \text{if } z < \mathrm{repr}_i(t) \text{ in } P, \\ > & \text{if } z > \mathrm{repr}_i(t) \text{ in } P, \\ \parallel & \text{if } z \parallel \mathrm{repr}_i(t) \text{ in } P, \\ = & \text{if } z = \mathrm{repr}_i(t), \\ * & \text{if } \mathrm{repr}_i(t) \text{ is undefined.} \end{cases}$$

Now, for a vertex $t \in V(T)$ we define

$$\mathrm{Active}(t) = \{\mathrm{vec}(z,t) \mid z \in P \text{ such that } \mathrm{root}(z) \text{ is below } t \text{ in } T\}.$$

Recall that above we allow $\mathrm{root}(z) = t$. Note that in general there are at most 5^{k+1} such possible vectors, so in particular for every $t \in V(T)$, we have $|\mathrm{Active}(t)| \leqslant 5^{k+1}$.

Next we define an auxiliary directed graph D accompanying our fixed tree-decomposition (together with a fixed coloring of P), that will play a key role in the remaining argument. The vertex set of D is $\bigcup_{t \in V(T)} D_t$, where $D_t = \{t\} \times \mathrm{Active}(t)$, and there is an edge from $(t, \mathrm{vec}(z,t))$ to $(t', \mathrm{vec}(z,t'))$ for every $t, t' \in V(T)$ with t being a parent of t' in T and every $z \in P$ with $\mathrm{root}(z)$ being below t in T.

Lemma 3. *For every edge $tt' \in E(T)$ with t being the parent of t' in T and every $d \in D_t$, the vertex d has exactly one out-neighbor in $D_{t'}$*

Proof. Let $t, t' \in V(T)$ with t being the parent of t'. To prove the lemma, we need to show that for every two distinct elements $x, y \in P$ with $\mathrm{root}(x)$ and $\mathrm{root}(y)$ being below t in T, and $\mathrm{vec}(x,t) = \mathrm{vec}(y,t)$, we have

$$\mathrm{vec}(x,t') = \mathrm{vec}(y,t').$$

To do so take an arbitrary $i \in [k+1]$. We distinguish three cases.

(i) $\mathrm{repr}_i(t')$ *is undefined.* Then by default we have $\mathrm{vec}_i(x,t') = * = \mathrm{vec}_i(y,t')$.
(ii) $\mathrm{repr}_i(t') = \mathrm{repr}_i(t)$ *are both defined and they are equal.* Then $\mathrm{vec}_i(x,t') = \mathrm{vec}_i(x,t) = \mathrm{vec}_i(y,t) = \mathrm{vec}_i(y,t')$.

(iii) $\mathrm{repr}_i(t')$ *is defined and* $\mathrm{repr}_i(t)$ *is undefined or both of them are defined but are different.* Let $z' = \mathrm{repr}_i(t')$. Then $t' = \mathrm{root}(z')$, in particular $z' \neq x, y$ and hence $\mathrm{vec}_i(x,t'), \mathrm{vec}_i(y,t') \in \{<, >, \|\}$. Suppose first that $\mathrm{vec}_i(x,t') = `<`$. This by definition means that $x < z'$ in P. Let us consider a cover chain of this relation in P. By the properties of a tree-decomposition this chain must contain an element z with $x \leqslant z < z'$ in P such that $z \in B_t \cap B_{t'}$. Then z is a representative of some color $i_0 \neq i$ both at t and at t', so by the previous case we know that $\mathrm{vec}_{i_0}(x,t') = \mathrm{vec}_{i_0}(y,t')$. This implies $y \leqslant z < z'$ in P and so we conclude $\mathrm{vec}_i(y,t') = `<`$ as required. Along the very same line, we can prove that if $\mathrm{vec}_i(x,t') = `>`$ then $\mathrm{vec}_i(y,t') = `>`$. However as $\mathrm{vec}_i(x,t')$ and $\mathrm{vec}_i(y,t')$ can only take three possible values, this already finishes the proof of this case.

□

We define a coloring $c_D : V(D) \to [5^{k+1}]$ as follows. Let $t_0, t_1, \ldots, t_m$ be the vertices of T in a topological order. Color all vertices from D_{t_0} with distinct colors. Now for $i \in [m]$ assuming that vertices in $\bigcup_{j<i} D_{t_j}$ are already colored we color D_{t_i} as follows:

(i) for every $d \in D_{t_i}$ such that d has at least one incoming edge in D we color d with the least color used on its in-neighbors;
(ii) once all vertices in D_{t_i} with at least one incoming edge are colored we color the remaining vertices in D_{t_i} with distinct colors not used so far on D_{t_i}.

Lemma 3 guarantees that for each $t \in V(T)$, the vertices in D_t all have distinct c_D-colors. Moreover, for every directed path F in D the c_D-colors of the vertices along the path are non-increasing. The decreasing sequence of c_D-colors we get after removing the repetitions is called the *signature* of F.

From now on, we will forget for a while about the underlying poset P, and we will only concentrate on exploring the dag D.

Let $d \in V(D)$ and let $c_D(d) = \gamma \in [5^{k+1}]$. Given d and γ, by Lemma 3, there is a unique directed subgraph of D which is a tree rooted at d and spans all the vertices of D that can be reached on a path from d in D with all vertices of the path c_D-colored with γ. Denote this tree by $\mathrm{tree}(d, \gamma)$.

For a subgraph D' of D, we define $\mathrm{proj}(D')$ to be a subgraph of T spanned by

$$\{t \in V(T) \mid D_t \cap D' \neq \emptyset\}.$$

Next, as a further preparation for our branching program, we construct inductively for every decreasing sequence Γ of colors from $[5^{k+1}]$ and every ternary sequence α over $\{0,1,2\}$ of length $|\alpha| = |\Gamma| - 1$ a family $\mathcal{F}_\Gamma^\alpha$ of subtrees of T. As a basis for this construction we have a family $\mathcal{F}_{(\gamma)}^\emptyset$ for each $\gamma \in [5^{k+1}]$. This family is defined to be

$$\mathcal{F}_{(\gamma)}^\emptyset = \mathcal{F}_\gamma = \{\mathrm{proj}(\mathrm{tree}(d,\gamma)) \mid d \text{ is a source in } D \text{ with } c_D(d) = \gamma\}.$$

An essential property of the families will be that the trees in $\mathcal{F}_\Gamma^\alpha$ are pairwise disjoint for every decreasing sequence Γ over $[5^{k+1}]$ and every ternary sequence α over $\{0,1,2\}$ with $|\alpha| = |\Gamma| - 1$.

Claim. *The family* $\mathcal{F}_{(\gamma)}^\emptyset$ *contains pairwise disjoint subtrees of* T *for every* $\gamma \in [5^{k+1}]$.

Proof. Suppose to the contrary that $\mathcal{F}_\gamma$ contains distinct subtrees Q_1, Q_2 of T with root vertices t_1, t_2, respectively, that share a vertex t. In particular t must be above both t_1

and t_2, implying that we either have t_1 below t_2 in T or the other way round. Without loss of generality, assume that we have t_1 below t_2 in T. For $i = 1, 2$ let $d_i \in D_{t_i}$ be such that $Q_i = \mathrm{proj}(\mathrm{tree}(d_i, \gamma))$. As t is in both Q_1 and Q_2, there must exist a vertex $d \in D_t$ of c_D-color γ, and this vertex has to belong to $D_i := \mathrm{tree}(d_i, \gamma)$ for $i = 1, 2$. In particular this implies that for $i = 1, 2$, there is a directed path from d_i to d in D_i with all vertices along the path being of c_D-color γ. As t_1 is below t_2 in T, the directed path from d_1 to d has to go through D_2; however, the only vertex there of c_D-color γ is d_2. This, unless $d_1 = d_2$ and hence $Q_1 = Q_2$, contradicts the fact that d_2 is a source vertex and so has no incoming edge. □

Now suppose that for a decreasing sequence Γ over $[5^{k+1}]$ and a ternary sequence α over $\{0, 1, 2\}$ with $|\alpha| = |\Gamma| - 1$, we are already given the family $\mathcal{F}^{\alpha}_{\Gamma}$. Let γ' be the last color in Γ, and let $\gamma \in [5^{k+1}]$ with $\gamma < \gamma'$. Then we define $\mathcal{F}^{\alpha 0}_{\Gamma\gamma} := \mathcal{F}_{\gamma}$. In particular, by the previous claim, we have that the subtrees in this family are pairwise disjoint.

To construct the two other families $\mathcal{F}^{\alpha 1}_{\Gamma\gamma}$ and $\mathcal{F}^{\alpha 2}_{\Gamma\gamma}$ we will use an intermediate family $\mathcal{F}^{\alpha +}_{\Gamma\gamma}$, which is produced by Algorithm 3. For the description of this algorithm we need additional notation. For a color $\gamma \in [5^{k+1}]$, we call a vertex t in T a *γ-break* if no vertex in D_t has c_D-color γ. For an edge tt' of T, t being the parent of t', and a color β with $\beta < \gamma$, we say that γ *merges into* β on tt' if

(i) there is a vertex $d \in D_t$ of color γ and a vertex $d' \in D_{t'}$ of color β, and
(ii) there is an edge from d to d' in D.

Finally, recall that for two vertices $u, v \in V(T)$ the unique path between them in T is denoted by $[u, v]$.

Algorithm 3: Construction of $\mathcal{F}^{\alpha +}_{\Gamma\gamma}$ given $\mathcal{F}^{\alpha}_{\Gamma}$ with $\Gamma \neq \emptyset$ and γ' being the last color of Γ

```
1  F^{α+}_{Γγ} ← ∅
2  for each Q in F^α_Γ do
3      Q^+ ← ∅,   r ← root of Q in T
4      for each t in Q do
5          for each t' child of t such that
6                  (a) t' is not in Q,
7                  (b) [r, t] contains a γ-break,
8                  (c) γ' merges into γ at tt' do
9              Q^+ ← Q^+ ∪ [r, t'] ∪ proj(tree(d', γ)),
10                 where d' is the vertex of color γ in D_{t'}
11     if Q^+ ≠ ∅ then
12         add Q^+ to F^{α+}_{Γγ}
13 return F^{α+}_{Γγ}
```

The following claim shows that $\mathcal{F}^{\alpha +}_{\Gamma\gamma}$ can be split into two families $\mathcal{F}^{\alpha 1}_{\Gamma\gamma}$ and $\mathcal{F}^{\alpha 2}_{\Gamma\gamma}$ such that each of them consists of pairwise disjoint trees.

Claim. *The family $\mathcal{F}^{\alpha +}_{\Gamma\gamma}$, produced by Algorithm 3, can be split into two parts, each consisting of pairwise disjoint subtrees of T.*

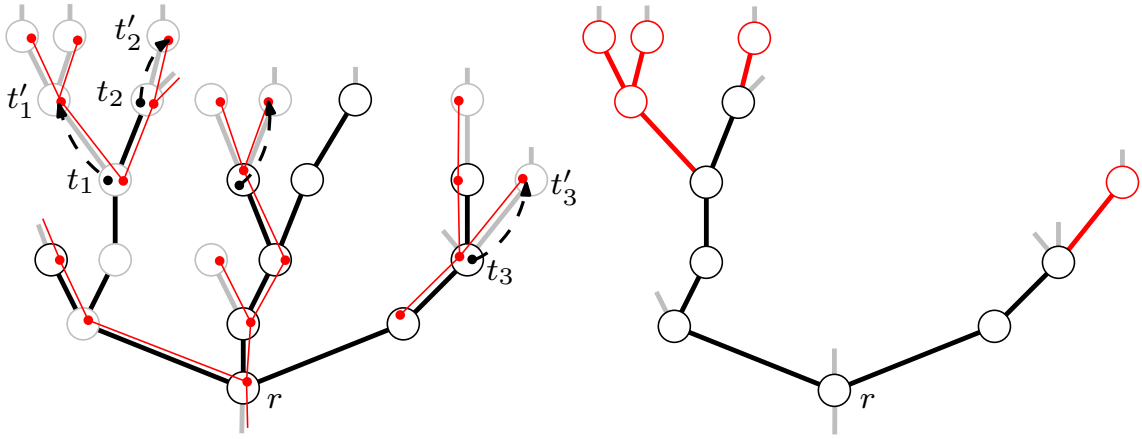

FIGURE 3. Iteration step of Algorithm 3. On the left Q is drawn with bold lines, the color γ is represented by red, and dashed arrows mark the places where γ' merges into γ. On the right Q^+ is drawn with its primal section in black and its extended section it red.

Proof. Every subtree $Q^+ \in \mathcal{F}^{\alpha+}_{\Gamma\gamma}$ produced by Algorithm 3 comes from some tree $Q \in \mathcal{F}^{\alpha}_{\Gamma}$. Based on this, we split each such Q^+ into two sections, the *primal section* $Q^+ \cap Q$ and the *extended section* $Q^+ \setminus Q$.

We show that for every two distinct trees $Q^+_1, Q^+_2 \in \mathcal{F}^{\alpha+}_{\Gamma\gamma}$ both their primal and their extended sections are disjoint, respectively. This is immediate for the primal sections, as those are subgraphs of some disjoint members of $\mathcal{F}^{\alpha}_{\Gamma}$.

Now consider the extended sections of Q^+_1, Q^+_2, and in order to get a contradiction, suppose that some vertex t is in both of them. For $i = 1, 2$ let Q_i be the pre-image of Q^+_i in $\mathcal{F}^{\alpha}_{\Gamma}$ with root vertex r_i. Note that r_i is also the root vertex of Q^+_i. Since for $i = 1, 2$, the vertex t is in the extended section of Q^+_i, there must be a vertex t_i of Q_i and a child t'_i of t_i such that $t \in \mathrm{proj}(\mathrm{tree}(d'_i, \gamma))$, where d'_i is the vertex of color γ in $D_{t'_i}$; moreover there is also a γ-break at some vertex t''_i on $[r_i, t_i]$. On the other hand, as members of $\mathcal{F}^{\alpha}_{\Gamma}$, the subtrees Q_1 and Q_2 are disjoint; hence $[r_1, t_1]$ and $[r_2, t_2]$ need to be disjoint intervals on the path from the root of T to t. Assuming without loss of generality that t_1 is below r_2 in T, we see that t''_2 is a γ-break in $[t_1, t]$, which contradicts the fact that $t \in \mathrm{proj}(\mathrm{tree}(d'_1, \gamma))$.

Hence, the intersection graph of the family $\mathcal{F}^{\alpha+}_{\Gamma}$ is a chordal graph (as every intersection graph of subtrees of a tree is chordal) with clique number two and so it is two-colorable. Now a two-fcoloring induces a partition of $\mathcal{F}^{\alpha+}_{i}$ into two families such that each consists of pairwise disjoint subtrees of T. □

The families $\mathcal{F}^{\alpha}_{\Gamma}$ form the base of the key subprogram of our branching program designed to prove Theorem 2.

Lemma 4. *For every decreasing color-sequence $\Gamma = (\gamma_1, \ldots, \gamma_\ell)$, there is a branching program $\mathcal{B}_\Gamma$, depending on at most $3^{\ell+1} - 1$ linear orders on $V(T)$, such that for every queried $x, y \in V(T)$ with $x \wedge y$ below x (resp. $x \wedge y$ below y) in T, $\mathcal{B}_\Gamma$ outputs 1 if and only if there is a path from $D_{x\wedge y}$ to D_x (resp. from $D_{x\wedge y}$ to D_y) in D with signature Γ.*

Proof. The two setups are clearly symmetric, so in the proof we concentrate only on the case when the queried vertices x, y are such that $x \wedge y$ is strictly below x in T.

For $1 \leqslant i \leqslant \ell$ let Γ_i denote the prefix of Γ of length i. The root node N_{root} of $\mathcal{B}_\Gamma$ starts with checking whether there is a vertex $d \in D_{x\wedge y}$ with c_D-color γ_1, which happens exactly if $x \wedge y$ is a vertex of some subtree $Q \in \mathcal{F}^{\emptyset}_{(\gamma_1)}$. This can be verified with the Color Detection Tool (Tool 2) using five linear orders. Indeed, color the edges of T as follows. For every $Q \in \mathcal{F}^{\emptyset}_{(\gamma_1)}$ and every vertex t of Q, color all edges going from t to its children in T with color Red and all other edges of T by Green. Then $x \wedge y$ is a vertex of some $Q \in \mathcal{F}^{\emptyset}_{(\gamma)}$ exactly if the first colored edge on $[x \wedge y, x]$ in T is Red. If the answer to this question is no, then N_{root} immediately returns 0, otherwise let d be the vertex in $D_{x\wedge y}$ with c_D -color γ_1. The branching program now proceeds step-by-step verifying whether the consecutive colors of the signature of the path from d to D_x (which by Lemma 3 is unique) agree with those in Γ.

For this we define for every $1 \leqslant i \leqslant \ell$ and $\alpha \in \{0, 1, 2\}^{i-1}$ a subprogram $N_{i,\alpha}$. The branching program is going to visit a sequence of these nodes starting with $N_{1,\emptyset}$. We always make sure that when $N_{i,\alpha}$ is visited, then the following invariant holds:

- Γ_i is a prefix of the signature of the path from d to D_x; moreover,
- if Γ_i is a *proper* prefix of the path from d to D_x then there is a tree $Q \in \mathcal{F}^{\alpha}_{\Gamma_i}$ such that $x \wedge y$ and t are both vertices of Q, where t is the unique vertex of T such that the maximal subpath of the path from d to D_x with signature Γ_i ends in D_t;
- otherwise when the signature of the path from d to D_x is equal to Γ_i, there is a tree $Q \in \mathcal{F}^{\alpha}_{\Gamma_i}$ such that $x \wedge y$ and x are both in Q.

Note that N_{root} already checked that $x \wedge y$ is contained in some tree $Q \in \mathcal{F}^{\emptyset}_{(\gamma_1)}$ that contains $\text{proj}(\text{tree}(d, \gamma_1))$ as a subtree, and hence the invariant holds for $i = 1$ and $\alpha = \emptyset$.

Now we continue with the description of $N_{i,\alpha}$ for $1 \leqslant i < \ell$ and $\alpha \in \{0, 1, 2\}^{i-1}$. This subprogram consists of two steps.

Step 1. The first task of $N_{i,\alpha}$ is to establish whether the next color on the path from d to D_x is γ_{i+1} or not. This can be done using five linear orders with Color Detection Tool (Tool 2). Indeed, color the edges of T as follows. For every $Q \in \mathcal{F}^{\alpha}_{\Gamma_i}$, every vertex t of Q and every child t' of t in T not belonging to Q, color the edge tt' Red if γ_i merges into γ_{i+1} at tt', otherwise color the edge tt' Green. Now, by the invariant that holds, the first colored edge on the path $[x \wedge y, x]$ is Red exactly if the path from d to D_x continues after its initial segment of signature Γ_i with color γ_{i+1}. Otherwise the path from d to D_x either changes to some different color after Γ_i or its signature is exactly Γ_i. In both cases N_{root} outputs 0.

Note that at this point if, the program is still running then we already know that Γ_{i+1} is a prefix of the signature of the path from d to D_x. In particular, Γ_i is a proper prefix. By the invariant, this implies that there is a tree $Q \in \mathcal{F}^{\alpha}_{\Gamma_i}$ such that $x \wedge y$ and t are both vertices of Q, where t is the unique vertex of T such that the maximal subpath of the path from d to D_x with signature Γ_i ends in D_t.

Step 2. The second task of $N_{i,\alpha}$ is to decide which subprogram to continue with, i.e., which is the relevant family out of $\mathcal{F}^{\alpha 0}_{\Gamma_{i+1}}$, $\mathcal{F}^{\alpha 1}_{\Gamma_{i+1}}$ and $\mathcal{F}^{\alpha 2}_{\Gamma_{i+1}}$.

The family $\mathcal{F}^{\alpha 0}_{\Gamma_{i+1}}$ is relevant if there is no γ_{i+1}-break on the path from $x \wedge y$ to the vertex t in Q. This again can be checked using five linear orders with the Color Detection

Tool (Tool 2). Indeed color the edges of T as follows. First for every $Q \in \mathcal{F}^{\alpha}_{\Gamma_i}$, every vertex t of Q and every child t' of t in T not belonging to Q, color the edge tt' GREEN. Then for each vertex $t \in V(T)$ which is a γ_{i+1}-break color all edges from t going to its children with RED (possibly overriding GREEN). Note that as $x \wedge y$ is in Q and x is not, we necessarily have a colored edge on the path $[x \wedge y, x]$, and the first colored edge on this path is GREEN exactly if there is no γ_{i+1}-break on $[x \wedge y, t]$. In this case, $N_{i+1,\alpha 0}$ is called next. Note that the required invariant holds by the construction of $\mathcal{F}^{\alpha 0}_{\Gamma_{i+1}}$.

Otherwise, if the first colored edge on $[x \wedge y, x]$ is RED, then there is an γ_{i+1}-break on the path $[x \wedge y, t]$. In this case we know that $\mathcal{F}^{\alpha +}_{\Gamma_i}$ contains a tree Q^+ which contains the projection of the maximal subpath of the path from d to D_x with signature Γ_{i+1}. Note that in particular, it also contains $x \wedge y$. Now it remains to decide whether this tree Q^+ belongs to $\mathcal{F}^{\alpha 1}_{\Gamma_{i+1}}$ or to $\mathcal{F}^{\alpha 2}_{\Gamma_{i+1}}$. Any tree in $\mathcal{F}^{\alpha +}_{i} = \mathcal{F}^{\alpha,1}_{i+1} \cup \mathcal{F}^{\alpha,2}_{i+1}$ has been produced as the offspring of some tree in $\mathcal{F}^{\alpha}_{\Gamma_i}$. Based on this, one can partition $\mathcal{F}^{\alpha}_{\Gamma_i}$ into three subfamilies: those with an offspring in $\mathcal{F}^{\alpha 1}_{\Gamma_{i+1}}$, those with an offspring in $\mathcal{F}^{\alpha 2}_{\Gamma_{i+1}}$ and the rest. Note that Q^+ is the offspring of the unique tree Q that contains $x \wedge y$, so with a repeated application of the Color Detection Tool (Tool 2), we can identify, each time using five linear orders, the subfamily (out of the first two) of $\mathcal{F}^{\alpha}_{\Gamma_i}$ containing Q. Indeed, first take the subfamily of $\mathcal{F}^{\alpha}_{\Gamma_i}$ containing the trees with an offspring in $\mathcal{F}^{\alpha 1}_{\Gamma_{i+1}}$. For every tree Q in this family and every vertex t of Q, color all edges going from t to its children in T with RED and color all other edges of T with GREEN. Now the first colored edge on $[x \wedge y, x]$ is RED exactly if $x \wedge y$ is in one of the trees from the first subfamily. After possibly repeating this for the second subfamily, we will know in which of them is Q, and based on that the program continues either with the subprogram $N_{i+1,\alpha 1}$ or with $N_{i+1,\alpha 2}$. Whichever is chosen, by the construction of $\mathcal{F}^{\alpha 1}_{\Gamma_{i+1}}$ and $\mathcal{F}^{\alpha 2}_{\Gamma_{i+1}}$ the required invariant holds.

This completes the description of the subprograms $N_{i,\alpha}$ for $1 \leqslant i < \ell$ and $\alpha \in \{0,1,2\}^{i-1}$. It remains to describe the finishing subprograms $N_{\ell,\alpha}$ with $\alpha \in \{0,1,2\}^{\ell-1}$. When $N_{\ell,\alpha}$ is called then by the invariant we know that Γ is the prefix of the signature of the path from d to D_x, and so the subprogram $N_{\ell,\alpha}$ needs to check whether the signature of this path is actually equal to Γ. Again by the invariant this happens exactly if $x \wedge y$ and x belong to the same tree in $\mathcal{F}^{\alpha}_{\Gamma}$. This can again be checked using the Color Detection Tool (Tool 2) using five linear orders. Color all edges of T that do not belong to any tree $Q \in \mathcal{F}^{\alpha}_{\Gamma}$ with color RED and keep all the other edges of T uncolored. Then the first edge on the path from $x \wedge y$ to x is RED exactly if $x \wedge y$ and x belong to the same tree in $\mathcal{F}^{\alpha}_{\Gamma}$. In case they do, N_{root} returns 1, otherwise it returns 0.

By construction the branching program $\mathcal{B}_\Gamma$ clearly returns 1 if and only if there is a path from $D_{x\wedge y}$ to D_x in D with signature Γ. What is still missing is to count the number of linear orders $\mathcal{B}_\Gamma$ is depending on. Note that every linear order that $\mathcal{B}_\Gamma$ uses comes from an application of the Color Detection Tool (Tool 2). The nodes N_{root} and $N_{\ell,\alpha}$, $\alpha \in \{0,1,2\}^{\ell-1}$ both involve one, while the nodes $N_{i,\alpha}$, $1 \leqslant i < \ell$, $\alpha \in \{0,1,2\}^{i-1}$ all involve four possible applications of Tool 2. In each of these applications two linear orders out of the total five are always the same, so the total number of linear orders appearing is at most

$$5 + \sum_{\alpha \in \{0,1,2\}^{\ell-1}} 3 + \sum_{i=1}^{\ell-1} \sum_{\alpha \in \{0,1,2\}^{i-1}} 4 \cdot 3 = 5 + 3^\ell + 4 \sum_{i=1}^{\ell-1} 3^i = 5 + 3^\ell + 4 \frac{3^\ell - 3}{3-1} = 3^{\ell+1} - 1.$$

□

4. The branching program for Theorem 2

Our aim here is to design a branching program to answer queries of the form "is $x \leqslant y$?" for elements $x, y \in P$. The branching program should depend on linear orders on the poset P and so we note that in the remaining part of this paper even though we will mostly construct linear orders on the vertices of the tree T we understand without saying that the corresponding linear order on P considered for the branching program is the one induced by the linear order restricted to the vertices of the form $\mathrm{root}(z)$ for some $z \in P$.

The root node N_{root} of this program first of all quickly separates the case $x = y$. This can be done with two initial linear orders on P where one is the reversal of the other. If $x \leqslant y$ in both linear orders then we must have $x = y$ and N_{root} outputs 1. Therefore, in what follows, we assume that the queried elements x and y are distinct.

The key ingredient behind the operation of the branching program is summarized in the following lemma. For $x, y \in P$ let $m_{xy} = \mathrm{root}(x) \wedge \mathrm{root}(y)$.

Lemma 5. *We have $x \leqslant y$ in P if and only if there exists*

- *a decreasing color sequence $\Gamma = (\gamma_1, \dots, \gamma_\ell)$ together with a path F_x in D such that it has signature Γ and $\mathrm{proj}(F_x) = [m_{xy}, \mathrm{root}(x)]$;*
- *a decreasing color sequence $\Delta = (\delta_1, \dots, \delta_m)$ with $\delta_1 = \gamma_1$ together with a path F_y in D such that it has signature Δ and $\mathrm{proj}(F_y) = [m_{xy}, \mathrm{root}(y)]$;*
- *a vertex $d_x = (\mathrm{root}(x), v^x) \in D_{\mathrm{root}(x)}$ such that $c_D(d_x) = \gamma_\ell$ and $v^x_{c(x)} = '>'$ or $'='$;*
- *a vertex $d_y = (\mathrm{root}(y), v^y) \in D_{\mathrm{root}(y)}$ such that $c_D(d_y) = \delta_m$ and $v^y_{c(y)} = '<'$ or $'='$.*

Proof. First suppose $x \leqslant y$ in P, and let $x = z_1 < z_2 < \cdots < z_s = y$ be a cover chain for this relation in P (note that here $s = 1$ is possible if $x = y$). By the basic properties of a tree decomposition, we know that the union of the subtrees of $z_1, \dots, z_s$ forms a subtree of T containing the path from $\mathrm{root}(x)$ to $\mathrm{root}(y)$. In particular, there must be an index $1 \leqslant i \leqslant s$ such that $z_i \in B_{m_{xy}}$. Then $\mathrm{root}(z_i)$ must be below m_{xy} in T, and so the vector $\mathrm{vec}(z_i, t)$ is defined for every vertex t above m_{xy}. Let F_x and F_y be the paths in D from $(m_{xy}, \mathrm{vec}(z_i, m_{xy}))$ to $d_x = (\mathrm{root}(x), \mathrm{vec}(z_i, \mathrm{root}(x)))$ and from $(m_{xy}, \mathrm{vec}(z_i, m_{xy}))$ to $d_y = (\mathrm{root}(y), \mathrm{vec}(z_i, \mathrm{root}(y)))$ containing only vertices of the form $(t, \mathrm{vec}(z_i, t))$. Then clearly $\mathrm{proj}(F_x) = [m_{xy}, \mathrm{root}(x)]$ and $\mathrm{proj}(F_y) = [m_{xy}, \mathrm{root}(y)]$. Now let $\Gamma = (\gamma_1, \dots, \gamma_\ell)$ and $\Delta = (\delta_1, \dots, \delta_m)$ be the signatures of F_x and F_y respectively. Here $\gamma_1 = \delta_1$ is just the c_D-color of $(m_{xy}, \mathrm{vec}(z_i, m_{xy}))$. To finish this direction, just note that from $x \leqslant z_i \leqslant y$, it follows that $\mathrm{vec}_{c(x)}(z_i, \mathrm{root}(x)) = '>'$ or $'='$ and $\mathrm{vec}_{c(y)}(z_i, \mathrm{root}(y)) = '<'$ or $'='$.

Now for the backwards implication, first note that as $\gamma_1 = \delta_1$ the initial vertex of the paths F_x and F_y has to be the same vertex in $D_{m_{xy}}$, let us denote it by $d = (m_{xy}, v)$. Now let $z \in P$ be such that $\mathrm{root}(z)$ is below m_{xy} in T and $v = \mathrm{vec}(z, m_{xy})$. Note that by Lemma 3, all vertices along the paths F_x and F_y are of the form $(t, \mathrm{vec}(z, t))$. In particular, we have $v^x = \mathrm{vec}(z, \mathrm{root}(x))$ and $v^y = \mathrm{vec}(z, \mathrm{root}(y))$. However, then $v^x_{c(x)} = '>'$ or $'='$ and $v^y_{c(y)} = '<'$ or $'='$ just mean that $x \leqslant z \leqslant y$ in P, as required. □

With Lemma 5 in place, we can continue with the description of the branching program. N_{root} will have a child $N_{\Gamma,\Delta}$ for every pair of decreasing color sequences Γ, Δ with the

same starting color, which will return 1 exactly if the requirements from Lemma 5 are satisfied with these color sequences. N_{root} then will call its children one-by-one until one of them returns a 1, in which case N_{root} also returns a 1. Otherwise, when all of its children return a 0, N_{root} also returns a 0.

What is left now is to describe how $N_{\Gamma,\Delta}$ works for given decreasing color sequences $\Gamma = (\gamma_1, \dots, \gamma_\ell)$ and $\Delta = (\delta_1, \dots, \delta_m)$ with $\gamma_1 = \delta_1$. It will have two children $N_{>,\Gamma}$ and $N_{<,\Delta}$, whose task will be to check the existence of the pairs (F_x, d_x) and (F_y, d_y) from Lemma 5, respectively, and so $N_{\Gamma,\Delta}$ will return a 1 exactly if both of its children succeed.

Because of symmetry, here we will only concentrate on the description of $N_{>,\Gamma}$, everything stated translates naturally to the setting of $N_{<,\Delta}$.

To start with, $N_{>,\Gamma}$ handles the question about the existence of d_x. This can be done using the Set Membership Tool (Tool 1) using three linear orders. Indeed, set

$$S_{\gamma_\ell} = \{z \in P \ \mid \ \exists\, d = (\text{root}(z), v) \in D_{\text{root}(z)} \text{ with } c_D(d) = \gamma_\ell \text{ and } v_{c(z)} = '>' \text{ or } '='\}.$$

Then d_x exists exactly if $x \in S_{\gamma_\ell}$. If this is not the case then $N_{>,\Gamma}$ returns 0; otherwise it continues with handling the question about the existence of F_x.

For this it next checks whether $m_{xy} = \text{root}(x)$ or m_{xy} is strictly below $\text{root}(x)$. This can be easily done by looking at the left-first-search and right-first-search order of the vertices of T.

Case 1 ($m_{xy} = \text{root}(x)$). Note that we already know that $D_{\text{root}(x)}$ contains a vertex of color γ_ℓ, so in this case a suitable F_x exists if and only if $\ell = 1$, so in this case $N_{x,\Gamma}$ returns a 1 if and only if $\ell = 1$.

Case 2 (m_{xy} is strictly below $\text{root}(x)$). In this case $N_{>,\Gamma}$ simply calls the subprogram $\mathcal{B}_\Gamma$ guaranteed by Lemma 4 for the vertices $\text{root}(x)$ and $\text{root}(y)$ and returns the same value as $\mathcal{B}_\Gamma$.

This finishes the description of the branching program. It is clear by the construction that it really returns 1 if and only if $x \leqslant y$ in P.

One final thing we need to do in order to prove Theorem 2 is to count how many linear orders does this branching program use. For this, first note that for fixed Γ (resp. fixed Δ) the child node $N_{>,\Gamma}$ (resp. $N_{<,\Delta}$) of the node $N_{\Gamma,\Delta}$ is the same for every Δ (resp. every Γ) – we think of them as being identical copies of the same node – and hence depend on the same set of linear orders, namely three linear orders related to the Set Membership Tool (Tool 1), the left-first-search and right-first-search order of the vertices of T (the same for every node) and the $3^{|\Gamma|+1} - 1$ (resp. $3^{|\Delta|+1} - 1$) linear orders related to the subprogram $\mathcal{B}_\Gamma$ (resp. $\mathcal{B}_\Delta$). In addition to this the root node uses two more linear orders

to identify the case $x = y$, which in total gives at most

$$\begin{aligned}
&2 + \sum_{\Gamma,\Delta:\delta_1=\gamma_1} \Big(3 + (3^{|\Gamma|+1} - 1) + 3 + (3^{|\Delta|+1} - 1)\Big) + 2 \\
&\leqslant 4 + \sum_{\Gamma}(3 + 3^{|\Gamma|+1} - 1) + \sum_{\Delta}(3 + 3^{|\Delta|+1} - 1) = 4 + 2\sum_{\ell=1}^{5^{k+1}} \binom{5^{k+1}}{\ell}(2 + 3^{\ell+1}) \\
&= 4 + 4\sum_{\ell=1}^{5^{k+1}} \binom{5^{k+1}}{\ell} + 6\sum_{\ell=1}^{5^{k+1}} 3^{\ell} = 4 + 4(2^{5^{k+1}} - 1) + 6((3+1)^{5^{k+1}} - 1) \\
&= 6 \cdot 4^{5^{k+1}} + 4 \cdot 2^{5^{k+1}} - 6.
\end{aligned}$$

linear orders, and so results $\mathrm{bdim}(P) \leqslant 6 \cdot 4^{5^{k+1}} + 4 \cdot 2^{5^{k+1}} - 6$. As the upper bound on $\mathrm{bdim}(P)$ only depends on k, this finishes the proof of Theorem 2.

5. A connection to reachability labeling schemes

Boolean realizers have a natural connection to labeling schemes for reachability queries for families of directed graphs. A similar application was already observed by Gambosi, Nešetřil and Talamo [7]. For a family $\vec{\mathcal{G}}$ of digraphs, a *labeling* is a non-negative integer function L that assigns for every digraph $\vec{G} \in \vec{\mathcal{G}}$ a *label* $L(v, \vec{G})$ to each vertex v of $\vec{G}$. A *reachability decoder* is a function f that given two labels λ_1, λ_2, returns a binary value $f(\lambda_1, \lambda_2)$. Now, a pair (L, f) is called a *reachability labeling scheme*, if for every digraph $\vec{G} \in \vec{\mathcal{G}}$ and every pair of vertices u, v in $\vec{G}$ there is a path from u to v in $\vec{G}$ if and only if $f(L(u, \vec{G}), L(v, \vec{G})) = 1$.

Formally we see labels as binary strings and given a label $L(v, \vec{G})$, we put $|L(v, \vec{G})|$ for its length as a binary string. Then the *size* of a reachability labeling scheme (L, f) (as a function of n) is

$$\max_{\vec{G}\in\vec{\mathcal{G}},|V(\vec{G})|=n} \max_{v\in V(\vec{G})} |L(v, \vec{G})|.$$

The central question in this area is to determine how small can reachability labeling schemes be for different families of graphs. A particularly interesting case is when $\vec{\mathcal{G}}$ is the family of planar digraphs. Along this line, Thorup [14] presented a reachability labeling scheme of size $O(\log^2 n)$ for planar digraphs on n vertices; however, it still remains a challenge to answer the following question.

Problem 6. *Is there a reachability labeling scheme of size $O(\log n)$ for planar digraphs on n vertices?*

We remark that a positive answer to a stronger version of Problem 1 would result, as argued in the observation below, in a positive answer for the above problem as well.

Observation. Let $\mathcal{G}$ be a family of (undirected) graphs which is closed under taking minors. Suppose that the boolean dimension of posets whose cover graph belongs to $\mathcal{G}$ is bounded from above by some constant k in a strong sense: the boolean formula participating in the boolean realizer can be chosen to be the formula ϕ for every poset. Note that in this case, the boolean realizers should contain exactly k linear orders for every poset. Then there exists a reachability labeling scheme of size $k \cdot \log n$ for the family $\vec{\mathcal{G}}$ of all digraphs whose undirected variant belongs to $\mathcal{G}$.

Indeed, to define the appropriate labeling, let $\vec{G}$ be and arbitrary digraph on n vertices from $\vec{\mathcal{G}}$. Note that for any reachability labeling, we may assume that vertices along a directed cycle all get the same label, so it is enough to deal with the acyclic digraph $\vec{G'}$ on $n' \leqslant n$ vertices, also belonging to $\vec{\mathcal{G}}$, that we get from $\vec{G}$ by contracting all directed cycles. Now that $\vec{G'}$ is acyclic, it gives rise to a poset P, whose elements are the vertices of $\vec{G'}$ and $x \leqslant y$ in P if and only if there is a directed path from x to y in $\vec{G'}$. So deciding whether $x \leqslant y$ in P is the same as deciding whether y is reachable from x in $\vec{G'}$. As the cover graph of P is the undirected version of $\vec{G'} \in \vec{\mathcal{G}}$, and as such belongs to $\mathcal{G}$, by assumption we have $\mathrm{bdim}(P) \leqslant k$. To finally define the labeling, take the k linear orders that together with ϕ form a boolean realizer for P, and for an element $x \in P = V(\vec{G'})$ put into $L(x, \vec{G'})$ the respective positions of x in the k linear orders. Then the size of any label is $k \cdot \log n' \leqslant k \cdot \log n$, and given two elements $x, y \in P = V(\vec{G'})$ from their labels, we can extract their relative position in the k linear orders, and, using ϕ, decide whether $x \leqslant y$ in P, i.e. whether there is a directed path from x to y in $\vec{G'}$.

Note that within the proof of Theorem 2, the formula (or the branching program) witnessing small boolean dimension does not depend on the poset P. Therefore, by the Observation above, we obtain a reachability labeling scheme of size $O(\log n)$ for digraphs on n vertices with bounded tree-width. As we pointed out this recently, such a reachability labeling scheme can be obtained more directly using the elimination tree of G witnessing the inequality $\mathrm{td}(G) \leqslant \mathrm{tw}(G) \log n$, where $\mathrm{td}(G)$ is the tree-depth of G.

6. Posets with unbounded boolean dimension whose cover graphs have maximum degree 3

A poset P is said to be an *interval order* if there is an assignment $P \ni x \to [\ell_x, r_x] \subseteq \mathbb{R}$ such that for every $x, y \in P$ we have $x < y$ in P if and only if $r_x < \ell_y$. This assignment is often called an *interval representation* of P. A *universal interval order* of order n ($n \geqslant 2$) is the poset U_n with elements $\{[i, j] \mid i, j \in \mathbb{N} \text{ and } 1 \leqslant i < j \leqslant n\}$ and $[i, j] < [k, \ell]$ in U_n if $j < k$. The proof of the following claim is an easy modification of an argument from [6].

Claim. $\mathrm{bdim}(U_n) \geqslant \log\log\log n$.

Proof. Consider a boolean realizer of U_n consisting of the linear orders $L_1, \ldots, L_d$ and a d-ary boolean formula ϕ. We will show that $d \geqslant \log\log\log n$.

The *double-shift graph* G_n of order n is the graph with vertex set $\{(i, j, k) \mid i, j \in \mathbb{N} \text{ and } 1 \leqslant i < j < k \leqslant n\}$ and (i, j, k) being adjacent to (j, k, ℓ) for all integers i, j, k, ℓ with $1 \leqslant i < j < k < \ell \leqslant n$. It is well known that $\chi(G_n) \geqslant \log\log n$, see e.g. [5].

For a triple of integers (i, j, k) with $1 \leqslant i < j < k \leqslant n$, we define $\psi(i, j, k)$ to be a binary sequence $(z_1, \ldots, z_d)$ of length d so that $z_\alpha = 1$ if $[i, j] < [j, k]$ in L_α, and $z_\alpha = 0$ if $[j, k] < [i, j]$ in L_α, for every $\alpha \in [d]$. We claim that ψ is a proper coloring of G_n. To see that, take four integers i, j, k, ℓ with $1 \leqslant i < j < k < \ell \leqslant n$ and suppose that $\psi(i, j, k) = (z_1, \ldots, z_d) = \psi(j, k, \ell)$. This means that for every $\alpha \in [d]$, if $z_\alpha = 1$ then $[i, j] < [j, k] < [k, \ell]$ in L_α, and if $z_\alpha = 0$ then $[k, \ell] < [j, k] < [i, j]$ in L_α. Since $[i, j]$ and $[j, k]$ are incomparable in U_n we have that $\phi(z_1, \ldots, z_d) = 0$. On the other hand, $[i, j] < [k, \ell]$ in U_n and therefore we must have $\phi(z_1, \ldots, z_d) = 1$, a contradiction. Thus

ψ is a proper coloring of G_n. Since ψ takes at most 2^d values we get that $2^d \geqslant \log\log n$, as desired. $\square$

It is easy to see that every interval order P has a *distinguishing* interval representation, which means an interval representation with all the endpoints ℓ_x, r_x for $x \in P$ being distinct.

Now consider an interval order P together with a distinguishing representation. Let $e_1, \ldots, e_m$ be the values of all endpoints in the representation in increasing order. Extend the poset P to P^+ by introducing new elements represented by the intervals $[e_i + \frac{1}{5}(e_{i+1} - e_i), e_i + \frac{2}{5}(e_{i+1} - e_i)]$, $[e_i + \frac{3}{5}(e_{i+1} - e_i), e_i + \frac{4}{5}(e_{i+1} - e_i)]$ for every $i \in [m-1]$. Note that every element of P^+ has at most three neighbors in the cover graph. Therefore, the maximum degree of the cover graph of P^+ is at most 3 and since P is a subposet of P^+ we have $\mathrm{bdim}(P^+) \geqslant \mathrm{bdim}(P)$.

This way we can also extend the universal interval order U_n to U_n^+, for every $n \geqslant 2$, and obtain a family of posets with cover graphs of maximum degree 3 and unbounded boolean dimension.

References

[1] Fidel Barrera-Cruz, Thomas Prag, Heather Smith, Libby Taylor, and William T. Trotter. Comparing Dushnik-Miller dimension, boolean dimension and local dimension. *Order*, 2019. https://doi.org/10.1007/s11083-019-09502-6.

[2] Bartłomiej Bosek, Jarosław Grytczuk, and William T. Trotter. Local dimension is unbounded for planar posets. submitted, arXiv:1712.06099.

[3] Graham R. Brightwell and Paolo Giulio Franciosa. On the Boolean dimension of spherical orders. *Order*, 13(3):233–243, 1996.

[4] Ben Dushnik and Edwin W. Miller. Partially ordered sets. *Amer. J. Math.*, 63:600–610, 1941.

[5] Stefan Felsner. *Interval Orders: Combinatorial Structure and Algorithms*. PhD thesis, Technische Universität Berlin, 1992.

[6] Z. Füredi, P. Hajnal, V. Rödl, and W. T. Trotter. Interval orders and shift graphs. In *Sets, graphs and numbers (Budapest, 1991)*, volume 60 of *Colloq. Math. Soc. János Bolyai*, pages 297–313. North-Holland, Amsterdam, 1992.

[7] G. Gambosi, J. Nešetřil, and M. Talamo. Posets, boolean representations and quick path searching. In Thomas Ottmann, editor, *Automata, Languages and Programming*, pages 404–424, Berlin, Heidelberg, 1987. Springer Berlin Heidelberg.

[8] Giorgio Gambosi, Jaroslav Nešetřil, and Maurizio Talamo. On locally presented posets. *Theoretical Computer Science*, 70(2):251 – 260, 1990.

[9] David Kelly. On the dimension of partially ordered sets. *Discrete Math.*, 35:135–156, 1981.

[10] Tamás Mészáros, Piotr Micek, and William T. Trotter. Boolean dimension, components and blocks. *Order*, 2019. https://doi.org/10.1007/s11083-019-09505-3.

[11] Piotr Micek and Walczak Bartosz. personal communication.

[12] J. Nešetřil and P. Pudlák. A note on Boolean dimension of posets. In *Irregularities of partitions (Fertőd, 1986)*, volume 8 of *Algorithms Combin. Study Res. Texts*, pages 137–140. Springer, Berlin, 1989.

[13] Michał Seweryn. Improved bound for the dimension of posets of treewidth two. *Discrete Mathematics*, 2019. https://doi.org/10.1016/j.disc.2019.111605.

[14] Mikkel Thorup. Compact oracles for reachability and approximate distances in planar digraphs. *J. ACM*, 51(6):993–1024, 2004.

[15] William T. Trotter, Jr. and John I. Moore, Jr. The dimension of planar posets. *J. Combinatorial Theory Ser. B*, 22(1):54–67, 1977.

[16] Torsten Ueckerdt. proposed at Order & Geometry workshop in Gułtowy, Poland, 2016.