Optimal Discretization is Fixed-parameter Tractable*

Stefan Kratsch† Tomáš Masařík‡ Irene Muzi§ Marcin Pilipczuk¶
Manuel Sorge‖

Abstract

Given two disjoint sets $W_1$ and $W_2$ of points in the plane, the **Optimal Discretization** problem asks for the minimum size of a family of horizontal and vertical lines that separate $W_1$ from $W_2$, that is, in every region into which the lines partition the plane there are either only points of $W_1$, or only points of $W_2$, or the region is empty. Equivalently, **Optimal Discretization** can be phrased as a task of discretizing continuous variables: we would like to discretize the range of $x$-coordinates and the range of $y$-coordinates into as few segments as possible, maintaining that no pair of points from $W_1 \times W_2$ are projected onto the same pair of segments under this discretization.

We provide a fixed-parameter algorithm for the problem, parameterized by the number of lines in the solution. Our algorithm works in time $2^{O(k^2 \log k)} n^{O(1)}$, where $k$ is the bound on the number of lines to find and $n$ is the number of points in the input.

Our result answers in positive a question of Bonnet, Giannopolous, and Lampis [IPEC 2017] and of Froese (PhD thesis, 2018) and is in contrast with the known intractability of two closely related generalizations: the **Rectangle Stabbing** problem and the generalization in which the selected lines are not required to be axis-parallel.

1 Introduction

For three numbers $a, b, c \in \mathbb{Q}$, we say that $b$ is between $a$ and $c$ if $a < b < c$ or $c < b < a$. The input to **Optimal Discretization** consists of two sets $W_1, W_2 \subseteq \mathbb{Q} \times \mathbb{Q}$ and an integer $k$. A pair $(X, Y)$ of sets $X, Y \subseteq \mathbb{Q}$ is called a separation (of $W_1$ and $W_2$) if for every $(x_1, y_1) \in W_1$ and $(x_2, y_2) \in W_2$ there exists an element of $X$ between $x_1$ and $x_2$ or an element of $Y$ between $y_1$ and $y_2$. In other words, we draw $|X|$ vertical lines at $x$-coordinates from $X$ and $|Y|$ horizontal lines at $y$-coordinates from $Y$ and focus on the $(|X| + 1)(|Y| + 1)$ regions the drawn lines partition the plane into. We require that the closure of every such region does not contain both a point from $W_1$ and a point from $W_2$. The optimization version of **Optimal Discretization** asks for a separation $(X, Y)$ minimizing $|X| + |Y|$; the decision version takes also an integer $k$ as an input and looks for a separation $(X, Y)$ with $|X| + |Y| \leq k$.

Looking at **Optimal Discretization** via the above geometric representation, one can also consider a variant where the lines are not required to be vertical or horizontal, but we want to draw a minimum number of lines such that every region into which the plane is partitioned by the drawn lines contains either only points of $W_1$, only points of $W_2$, or is empty. Bonnet, Giannopolous, and Lampis [1] studied this variant, denoted **Red-Blue Points Separation** with the points of $W_1$ being red and points of $W_2$ being blue, and

---

*This research is a part of a project that have received funding from the European Research Council (ERC) under the European Union’s Horizon 2020 research and innovation programme Grant Agreement 714704 (TM, IM, MP, and MS) and 648527 (IM). TM is also supported by Charles University, student grant number SVV–2017–260452.

†Institut für Informatik, Humboldt-Universität zu Berlin, Germany, kratsch@informatik.hu-berlin.de
‡Department of Applied Mathematics, Charles University, Czech Republic & Faculty of Mathematics, Informatics and Mechanics, University of Warsaw, Poland, masarik@kam.mff.cuni.cz
§Technische Universität Berlin, Germany, irene.muzi@tu-berlin.de
¶Faculty of Mathematics, Informatics and Mechanics, University of Warsaw, Poland, malcin@mimuw.edu.pl
‖Faculty of Mathematics, Informatics and Mechanics, University of Warsaw, Poland, manuel.sorge@mimuw.edu.pl
proved it is \( \mathsf{W}[1] \)-hard when parameterized by the number of lines. They conjectured that the \textsc{Optimal Discretization} problem (in their language, the \textit{axis-parallel} variant of \textsc{Red-Blue Points Separation}) is fixed-parameter tractable when parameterized by \( k \), the number of lines, but were able only to prove tractability under a weaker parameterization, namely the cardinality of the smaller of the sets \( W_1 \) and \( W_2 \).

\textsc{Optimal Discretization} has applications in machine learning, as an abstraction of a task of discretizing continuous variables \cite{8,10} (this motivation also is the source of the name of the problem). We would like to discretize the range of \( x \)-coordinates and the range of \( y \)-coordinates into as few segments as possible, maintaining that no pair of points from \( W_1 \times W_2 \) are projected onto the same pair of segments under this discretization. Within this language, fixed-parameter tractability of \textsc{Optimal Discretization} was posed as an open question by Froese \cite[Section 5.5]{9}.

In this work we establish fixed-parameter tractability of \textsc{Optimal Discretization} by showing the following.

\textbf{Theorem 1.1.} \textsc{Optimal Discretization} can be solved in time \( 2^{O(k^2 \log k)} n^{O(1)} \).

\textsc{Optimal Discretization} is a special case of the \textsc{Rectangle Stabbing} problem, where we are given a set of axis-parallel rectangles in the plane and the goal is to draw \( k \) horizontal or vertical lines that intersect all of the input rectangles. \textsc{Rectangle Stabbing} is \( \mathsf{W}[1] \)-hard when parameterized by the number of lines \cite{7} even if all the rectangles are squares of the same size. That, together with the hardness of \textsc{Red-Blue Points Separation} (with lines not restricted to axis-parallel ones) make the tractability result of Theorem 1.1 slightly unexpected.

The basic approach we use in proof of Theorem 1.1 is as follows. Let \((X_0, Y_0)\) be an approximate solution (that can be obtained via e.g. the iterative compression technique or a known polynomial-time 2-approximation algorithm \cite{4}). If we know that there exists an optimal solution \((X, Y)\) such that between every two consecutive elements of \( X_0 \) there is at most one element of \( X \) and between every two consecutive elements of \( Y_0 \) there is at most one element of \( Y \), we can proceed as follows.

First, for every two consecutive elements of \( X_0 \) we guess (trying both possibilities) whether there is an element of \( X \) between them and similarly for every two consecutive elements of \( Y_0 \). This gives us a general picture of the layout of the lines of \( X, X_0, Y, \) and \( Y_0 \).

Consider all \( O(k^2) \) cells in which the vertical lines with \( x \)-coordinates from \( X_0 \cup X \) and the horizontal lines with \( y \)-coordinates from \( Y_0 \cup Y \) partition the plane. Similarly, consider all \( O(k^2) \) supercells in which the vertical lines with \( x \)-coordinates from \( X_0 \) and the horizontal lines with \( y \)-coordinates from \( Y_0 \) partition the plane. Every cell is contained in exactly one supercell. For every cell, guess whether it is empty or contains a point of \( W_1 \cup W_2 \). Note that the fact that \((X_0, Y_0)\) is a solution implies that every supercell contains only points from \( W_1 \), only points from \( W_2 \), or is empty. Hence, for each nonempty cell we can deduce whether it contains only points of \( W_1 \) or only points of \( W_2 \). Check Figure \ref{fig:1} for an example of such a situation.

We treat every element of \( X \cup Y \) as a variable with a domain being all rationals between the closest lines of \( X_0 \) or \( Y_0 \), respectively.

Now, the assumption that between every two consecutive elements of \( X_0 \) there is at most one element of \( X \) and similarly for \( Y \) and \( Y_0 \) ensures that every cell has at most two borders coming from \( X \cup Y \). Thus, for every cell \( C \) that is guessed to be empty and every point \( p \) in the supercell containing \( C \) we add a constraint binding the at most two borders of \( C \) from \( X \cup Y \), asserting that \( p \) does not land in \( C \).

The crucial observation is that the CSP instance constructed in this manner admits the median as a so-called majority polymorphism and such CSPs are polynomial-time solvable (for more on majority polymorphisms, which are ternary near-unanimity polymorphisms, see e.g. \cite{2} or \cite{5}).

The above approach breaks down if there are multiple lines of \( X \) between two consecutive elements of \( X_0 \). One can still construct a CSP instance with variables corresponding to the lines of \( X \cup Y \) and constraints asserting that the content of the cells is as we guessed it to be. However, it is possible to show that the constructed CSP instance no longer admits a majority polymorphism.

To cope with that, we perform an involved series of branching and color-coding steps on the instance to clean up the structure of the constructed constraints and obtain a tractable CSP instance. We were not able
Figure 1: Example of a basic situation. An approximate solution \((X_0, Y_0)\) is denoted by solid lines, an optimal solution \((X, Y)\) by dashed lines. A supercell is marked by green color and a cell by orange color.

to reduce to a known tractable case; instead, in Section 3 we introduce a special CSP variant and prove its tractability via a nontrivial branching algorithm.

2 Segments, segment reversions, and segment representations

2.1 Basic definitions and observations

**Definition 2.1.** For a finite totally ordered set \((D, \leq)\) and two elements \(x, y \in D, x \leq y\), the **segment** between \(x\) and \(y\) is
\[ D[x, y] = \{ z \in D \mid x \leq z \leq y \} \]
Elements \(x\) and \(y\) are the **endpoints** of the segment \(D[x, y]\).

We often write just \([x, y]\) for the segment \(D[x, y]\) if the set \((D, \leq)\) is clear from the context.

**Definition 2.2.** Let \((D, \leq)\) be a finite totally ordered set and let \(D = \{a(1), a(2), \ldots, a(|D|)\}\) with \(a(i) < a(j)\) iff \(i < j\).

A permutation \(\pi : D \to D\) is a **segment reversion** of \(D\) if there exist integers \(1 = i_1 < i_2 < \ldots < i_\ell = |D| + 1\) such that for every \(j \in [\ell]\) and every integer \(x\) with \(i_j \leq x < i_{j+1}\) we have \(\pi(a(x)) = a(i_{j+1} - 1 - (x - i_j))\). In other words, a segment reversion is a permutation that partitions the domain \(D\) into segments \([a(i_1), a(i_2-1)], [a(i_2), a(i_3-1)], \ldots, [a(i_\ell), a(i_{\ell}-1)]\) and reverses every segment independently.

A **segment representation of depth** \(k\) of a permutation \(\pi\) of \(D\) is a sequence of \(k\) segment reversions \(\pi_1, \pi_2, \ldots, \pi_k\) of \(D\) such that their composition satisfies \(\pi = \pi_k \circ \pi_{k-1} \circ \ldots \circ \pi_1\). A permutation \(\pi : D \to D\) is of **depth** at most \(k\) if \(\pi\) admits a segment representation of depth at most \(k\).

A **segment representation of depth** \(k\) of a function \(\phi : D \to \mathbb{N}\) is a tuple of \(k\) segment reversions \(\pi_1, \pi_2, \ldots, \pi_k\) of \(D\) and a nondecreasing function \(\phi'\) such that their composition satisfies \(\phi = \phi' \circ \pi_1 \circ \pi_2 \circ \ldots \circ \pi_k\).

**Definition 2.3.** Let \((D, \leq)\) be a finite totally ordered set. A **segment partition** is a family \(\mathcal{P}\) of segments of \((D, \leq)\) which is a partition of \(D\). If for two segment partitions \(\mathcal{P}_1\) and \(\mathcal{P}_2\) we have that for every \(P_1 \in \mathcal{P}_1\) there exists \(P_2 \in \mathcal{P}_2\) with \(P_1 \subseteq P_2\) then we say that \(\mathcal{P}_1\) is **more refined than** \(\mathcal{P}_2\) or \(\mathcal{P}_2\) is **coarser than** \(\mathcal{P}_1\).

The notion of a coarser partition turns the family of all segment partitions into a partially ordered set with two extremal values, the **most coarse** partition with one segment and the **most refined** partition with all segments being singletons.
Note that every segment partition $\mathcal{P}$ induces a segment reversion that reverses the segments of $\mathcal{P}$. We will denote this segment reversion as $g_\mathcal{P}$.

**Definition 2.4.** Let $(D_i, \leq_i)$ for $i = 1, 2$ be two finite totally ordered sets.

A relation $R \subseteq D_1 \times D_2$ is **downwards-closed** if for every $(a, b) \in R$ and $a' \leq_1 a$, $b' \leq_2 b$ it holds that $(a', b') \in R$.

A relation $R \subseteq D_1 \times D_2$ is of **depth at most $k$** if there exists a permutation $\pi_1$ of $D_1$ of depth at most $k_1$, a permutation $\pi_2$ of $D_2$ of depth at most $k_2$, and a downwards-closed relation $R' \subseteq D_1 \times D_2$ such that $k_1 + k_2 \leq k$ and $(a, b) \in R$ if and only if $(f_1(a), f_2(b)) \in R'$. A **segment representation** of $R$ consists of $R'$, a segment representation of $\pi_1$ of depth at most $k_1$ and a segment representation of $\pi_2$ of depth at most $k_2$.

We make two straightforward observations regarding some relations that are of small depth.

**Observation 2.5.** Let $(D_1, \leq_1)$ and $(D_2, \leq_2)$ be two finite totally ordered sets. For $i = 1, 2$, let $(a_i^j)_{j=1}^2$ be a sequence of elements of $D_i$. Then a relation $R \subseteq D_1 \times D_2$ defined as $(x_1, x_2) \in R$ if and only if:

- $\bigvee_{j=1}^2 (x_1 \leq_1 a_1^j) \lor (x_2 \leq_2 a_2^j)$ is downwards-closed and thus of depth 0;
- $\bigvee_{j=1}^2 (x_1 \leq_1 a_1^j) \lor (x_2 \geq_2 a_2^j)$ is of depth 1, using $k_1 = 0$ and $k_2 = 1$ and a segment reversion with one segment reversing the whole $D_2$;
- $\bigvee_{j=1}^2 (x_1 \geq_1 a_1^j) \lor (x_2 \leq_2 a_2^j)$ is of depth 1, using $k_1 = 1$ and $k_2 = 0$ and a segment reversion with one segment reversing the whole $D_1$;
- $\bigvee_{j=1}^2 (x_1 \geq_1 a_1^j) \lor (x_2 \geq_2 a_2^j)$ is of depth 2, using $k_1 = 1$ and $k_2 = 1$ and segment reversions each with one segment reversing the whole $D_1$ and the whole $D_2$, respectively.

Thus, a conjunction of an arbitrary finite number of the above relations can be expressed as a conjunction of at most four relations, each of depth at most 2.

**Observation 2.6.** Let $D_1, D_2 \subseteq D$ for a totally ordered set $(D, \leq)$. We treat $D_1$ as a totally ordered set with the order inherited from $(D, \leq)$. Then a relation $R \subseteq D_1 \times D_2$ defined as $R = \{(x_1, x_2) \in D_1 \times D_2 \mid x_1 < x_2\}$ is of depth at most 1 and a segment representation of this depth can be computed in polynomial time.

**Proof.** Let $\pi_2$ be a segment reversion of $D_2$ with one segment, that is, $\pi_2$ reverses the domain $D_2$. Observe that $\{(a, \pi_2(b)) \mid a \in D_1 \land b \in D_2 \land a < b\}$ is a downwards-closed subrelation of $D_1 \times D_2$.

### 2.2 Operating on segment representations

We will need the following two technical lemmata.

**Lemma 2.7.** Let $(D_1, \leq_1)$ and $(D_2, \leq_2)$ be two finite totally ordered sets, $f : D_1 \to D_2$ be a nondecreasing function\(^1\) and $g : D_2 \to D_2$ be a segment reversion. Then there exists a nondecreasing function $f' : D_1 \to D_2$ and a segment reversion $g' : D_2 \to D_1$ such that $g \circ f = f' \circ g'$. Furthermore, such $f'$ and $g'$ can be computed in polynomial time, given $(D_1, \leq_1)$, $(D_2, \leq_2)$, $f$, and $g$.

**Proof.** Let $(D_2[a_i, b_i])_{i=1}^r$ be the segments of the segment reversion $g$ in increasing order. For every $i \in \{1, 2, \ldots, r\}$, let

$$c_i = \min\{e \in D_1 \mid f(e) \geq a_i\},$$

$$d_i = \max\{d \in D_1 \mid f(d) \leq b_i\}.$$

Let $Q$ be the family of those segments $D_1[c_i, d_i]$ for which both $c_i$ and $d_i$ are defined and $c_i \leq_1 d_i$ (which is equivalent to the existence of $x \in D_1$ with $f(x) \in D_2[a_i, b_i]$). From the definition of $c_i$s and $d_i$s we obtain that $Q$ is a segment partition of $(D_1, \leq_1)$. We put $g' = g_{Q}$ and

$$f' = g \circ f \circ g'.$$

\(^{1}\)Throughout, for some relation $\leq$ we use $x < y$ to denote $x \leq y$ and not $x = y$.

\(^{2}\)A function $f$ on a domain and codomain that are totally ordered by $\leq_1$ and $\leq_2$, respectively, is called **nondecreasing** if for every $x, x'$ in the domain we have that $x \leq_1 x'$ implies $f(x) \leq_2 f(x')$. 

4
The desired equation $g \circ f = f' \circ g'$ follows directly from the definition of $f'$ and the fact that the segment reversion $g'$ is an involution. Clearly, $f'$ and $g'$ are computable in polynomial time. It remains to check that $f'$ is nondecreasing.

Let $x <_1 y$ be two elements of $D_1$. We consider two cases. In the first case, we assume that $x$ and $y$ belong to the same segment $D_1[c_i, d_i]$ of $Q$. Then, $g'(x)$ and $g'(y)$ also lie in $D_1[c_i, d_i]$ and $g'(x) \geq_1 g'(y)$ by the definition of the segment reversion $g' = g_2$. Since $f$ is nondecreasing, $f(g'(x)) \geq_2 f(g'(y))$. By the definition of $c_i$ and $d_i$, we have that both $f(g'(x))$ and $f(g'(y))$ lie in the segment $D_2[a_i, b_i]$. Hence, since $D_2[a_i, b_i]$ is a segment of the segment reversion $g$, we have $g(f(g'(x))) \leq_2 g(f(g'(y)))$, as desired.

In the second case, let $x \in D_1[c_i, d_i]$ and $y \in D_1[c_j, d_j]$ for some $i \neq j$. From the definition of the $c_i$s and $d_i$s we infer that $x <_1 y$ implies $i < j$. By the definition of $g' = g_2$, we have $g'(x) \in D_1[c_i, d_i]$ and $g'(y) \in D_1[c_j, d_j]$. Since $f$ is nondecreasing, $f(g'(x)) \leq_2 f(g'(y))$. By the definition of the $c_i$s and $d_i$s, we have that $f(g'(x)) \in D_2[a_i, b_i]$ and $f(g'(y)) \in D_2[a_j, b_j]$. Since $D_2[a_i, b_i]$ and $D_2[a_j, b_j]$ are segments of $g$, we have $g(f(g'(x))) \leq_2 g(f(g'(y)))$, as desired.

This finishes the proof that $f'$ is nondecreasing and concludes the proof of the claim.

**Lemma 2.8.** Let $(D_i, \leq_i)$ for $i = 1, 2, 3$ be three finite totally ordered sets, $f : D_1 \to D_2$ be a nondecreasing function, and $R \subseteq D_2 \times D_3$ be a downwards-closed relation. Then the relation

$$R' = \{(x, y) \in D_1 \times D_3 \mid (f(x), y) \in R\}$$

is also downwards-closed.

**Proof.** If $(x, y) \in R'$, $x' \leq_1 x$, and $y' \leq_2 y$, then $f(x') \leq_2 f(x)$ as $f$ is nondecreasing, $(f(x'), y') \in R$ as $(f(x), y) \in R$ and $R$ is downwards closed, and thus $(x', y') \in R'$ by the definition of $R'$.

### 2.3 Tree of segment partitions

For a rooted tree $T$, we use the following notation:

- **leaves**($T$) is the set of leaves of $T$;
- **root**($T$) is the root of $T$;
- for a non-root node $v$, **parent**($v$) is the parent of $v$.

In this subsection we are interested in the following setting. A **tree of segment partitions** consists of:

- a finite totally ordered set $(D, \leq)$;
- a rooted tree $T$;
- a segment partition $P_v$ of $(D, \leq)$ for every $v \in V(T)$ such that:
  - the partition $P_{\text{parent}(v)}$ is coarser than the partition $P_v$ for every non-root node $v$;
  - the partition $P_{\text{root}(v)}$ is the most coarse partition (with one segment);
  - for every leaf $v \in \text{leaves}(T)$ the partition $P_v$ is the most refined partition (with only singletons);
- an assignment type : $V(T) \setminus \{\text{root}(T)\} \to \{\text{inc}, \text{dec}\}$.

We say that a non-root node $w$ is of **increasing type** if type($w$) = inc and of **decreasing type** if type($w$) = dec.

Given a tree of segment partitions $T = ((D, \leq), T, (P_v)_{v \in V(T)}, \text{type})$, a **family of leaf functions** is a family $(f_v)_{v \in \text{leaves}(T)}$ such that for every $v \in \text{leaves}(T)$ the function $f_v : D \to \mathbb{Z}$ satisfies the following property: for every non-root element $w$ on the path in $T$ from $v$ to $\text{root}(T)$, for every $Q \in P_{\text{parent}(w)}$, if $Q_1, Q_2, \ldots, Q_a$ are the segments of $P_w$ contained in $Q$ in increasing order, then for every $x_1 \in Q_1, x_2 \in Q_2, \ldots, x_a \in Q_a$ we have

$$f_v(x_1) < f_v(x_2) \ldots < f_v(x_a) \quad \text{ if type}(w) = \text{inc},$$

$$f_v(x_1) > f_v(x_2) \ldots > f_v(x_a) \quad \text{ if type}(w) = \text{dec}.$$  

\footnote{An involution is a function $\phi$ which is its own inverse, that is, $\phi \circ \phi$ is the identity.}
Lemma 2.9. Let $T = ((D, \leq), T, (P_v)_{v \in V(T), \text{type}})$ be a tree of segment partitions and $F = (f_v)_{v \in \text{leaves}(T)}$ be a family of leaf functions in $T$. Then there exists a family $\mathcal{G} = (g_v)_{v \in V(T) \setminus \{\text{root}(T)\}}$ of segment reversions of $D$ and a family $\hat{F} = (\hat{f}_v)_{v \in \text{leaves}(T)}$ of strictly increasing functions with domain $D$ and range $\mathbb{Z}$ such that, for every $v \in \text{leaves}(T)$, if $v = v_1, v_2, \ldots, v_b = \text{root}(T)$ are the nodes on the path from $v$ to $\text{root}(T)$ in $T$, then

$$f_v = \hat{f}_v \circ g_{v_{b-1}} \circ g_{v_{b-2}} \circ \ldots \circ g_{v_1}.$$ \hspace{1cm} (1)

Furthermore, given $T$ and $F$, the families $\mathcal{G}$ and $\hat{F}$ can be computed in polynomial time.

Proof. Fix a non-root node $w$. We say that $w$ is pivotal if either

- parent($w$) = root($T$) and type($w$) = dec, or
- parent($w$) \neq root($T$) and type($w$) \neq type(parent($w$)).

Let $Q_w = P_{\text{parent}(w)}$ if $w$ is pivotal and let $Q_w$ be the most refined partition of $D$ otherwise. Let $g_w = g_{Q_w}$. That is, $g_w$ is the segment reversion that reverses the segments of $P_{\text{parent}(w)}$ for pivotal $w$ and is an identity otherwise.

Fix a leaf $v \in \text{leaves}(T)$ and let $v = v_1, v_2, \ldots, v_b = \text{root}(T)$ be the nodes on the path in $T$ from $v$ to the root $\text{root}(T)$. Define

$$\hat{f}_v = f_v \circ g_{v_1} \circ g_{v_2} \circ \ldots \circ g_{v_{b-1}}.$$ Clearly, as a segment reversion is an involution, [1] follows. Hence, to finish the proof of the lemma it suffices to show that $\hat{f}_v$ is strictly increasing.

Take $x, y \in D$ with $x < y$. For each $i \in \{1, 2, \ldots, b\}$, let

$$x_i = g_{v_i} \circ g_{v_{i+1}} \circ \ldots \circ g_{v_{b-1}}(x),$$

$$y_i = g_{v_i} \circ g_{v_{i+1}} \circ \ldots \circ g_{v_{b-1}}(y),$$

and let $x_b = x$ and $y_b = y$. Recall that $P_{v_b} = P_{\text{root}(T)}$ is the most coarse partition with only one segment so $x_b, y_b$ lie in the same segment of $P_{v_b}$. Let $\ell \leq b$ be the minimum index such that $x_{\ell}$ and $y_{\ell}$ lie in the same segment of $P_{v_{\ell}}$. Note that $\ell > 1$ as $P_{v_1} = P_v$ is the most refined partition with singletons only. For each $i \in \{\ell, \ell + 1, \ldots, b\}$, let $Q_i \in P_{v_i}$ be the segment containing $x_i$ and $y_i$. Observe that, since $Q_i$, is a more refined partition than $P_{v_i}$, for every $i \in \{\ell, \ell + 1, \ldots, b\}$, elements $x_i$ and $y_i$ lie in the same segment of the partition $P_{v_i}$.

From the definition of being pivotal it follows that the number of indices $j \in \{\ell, \ell + 1, \ldots, b\}$ for which $v_{j-1}$ is pivotal is odd if type$(v_{j-1}) = \text{dec}$ and even if $\text{type}(v_{j-1}) = \text{inc}$. Recall that $g_{j-1}$ reverses the segment containing $x_{j-1}$ and $y_{j-1}$ if and only if $v_{j-1}$ is pivotal. Hence $x_{\ell-1} < y_{\ell-1}$ if $\text{type}(v_{\ell-1}) = \text{inc}$ and $x_{\ell-1} > y_{\ell-1}$ if $\text{type}(v_{\ell-1}) = \text{dec}$.

Since for every $i \in \{1, 2, \ldots, \ell\}$, we have that $x_i$ and $y_i$ lie in different segments of $P_{v_i}$, we have $x_1 < y_1$ if $\text{type}(v_{\ell-1}) = \text{inc}$ and $x_1 > y_1$ if $\text{type}(v_{\ell-1}) = \text{dec}$. For the same reason, $x_1$ and $y_1$ lie in different segments of $P_{v_1}$. From the definitions of increasing and decreasing types, we infer that if $\text{type}(v_{\ell-1}) = \text{inc}$, then $f_v(x_1) < f_v(y_1)$ as $x_1 < y_1$ and if $\text{type}(v_{\ell-1}) = \text{dec}$, then $f_v(x_1) < f_v(y_1)$ as $x_1 > y_1$. Observe that $f_v(x) = f_v(x_1)$ and $f_v(y) = f_v(y_1)$. Thus, in both cases, we obtain that $\hat{f}_v(x) < \hat{f}_v(y)$, as desired. \hfill \Box

3 Auxiliary CSP

In this section we will be interested in checking the satisfiability of the following constraint satisfaction problem (CSP).

**Definition 3.1.** An auxiliary CSP instance is a tuple $(\mathcal{X}, \mathcal{D}, \mathcal{C})$ consisting of a set $\mathcal{X} = \{x_1, x_2, \ldots, x_k\}$ of $k$ variables, a totally ordered finite domain $(D_i, \leq_i) \in \mathcal{D}$ for every variable $x_i$, and a set $\mathcal{C}$ of binary constraints. Each constraint $C \in \mathcal{C}$ is a tuple $(x_{i(C,1)}, x_{i(C,2)}, R_C)$ consisting of two variables $x_{i(C,1)}$ and $x_{i(C,2)}$, and a relation $R_C \subseteq D_{i(C,1)} \times D_{i(C,2)}$ given as a segment representation of some depth. We say that
constraint $C$ binds $x_i(c,1)$ and $x_i(c,2)$. An assignment is a function $\phi : X \to D$ such that for each $x_i \in X$ we have $\phi(x_i) \in D_i$. An assignment $\phi$ is satisfying if for each constraint $C = (x_i(c,1), x_i(c,2), R_C) \in \mathcal{C}$ we have $(\phi(x_i(c,1)), \phi(x_i(c,2))) \in R_C$.

Qualitatively, the main result of this section is the following.

**Theorem 3.2.** Checking satisfiability of an auxiliary CSP instance is fixed-parameter tractable when parameterized by the sum of the number of variables, the number of constraints, and the depths of all segment representations of constraints.

To prove Theorem 3.2 we show a more general result stated in Lemma 3.4 below. For this and to state precisely the running time bounds of the obtained algorithm, we need a few extra definitions. For a forest $F$, $\text{trees}(F)$ is the family of trees (connected components) of $F$. For $y \in V(F)$, $\text{tree}_F(y)$ is the tree of $F$ that contains $y$. We omit the subscript if it is clear from the context.

**Definition 3.3.** A forest-CSP instance is a tuple consisting of a forest $F$ with its vertex set $V(F)$ being the set of variables of the instance, an ordered finite domain $(D_T, \leq_T)$ for every $T \in \text{trees}(F)$ (that is, one domain shared between all vertices of $T$), for every $e \in E(T)$ and $T \in \text{trees}(F)$ a segment reversion $g_e$ that is a segment reversion of $D_T$, and a family of constraints $\mathcal{C}$. Each constraint $C \in \mathcal{C}$ is a tuple $(y_1, y_2, R_C)$ where $y_1, y_2 \in V(F)$ are variables and $R_C \subseteq D_{\text{tree}(y_1)} \times D_{\text{tree}(y_2)}$ is a downwards-closed relation. We say that $C$ binds $y_1$ and $y_2$.

An assignment is a function $\phi : V(F) \to D$ such that for each $y \in V(F)$ we have $\phi(y) \in D_{\text{tree}(y)}$. An assignment $\phi$ satisfies the forest-CSP instance if for every edge $yy' \in E(F)$ we have $g_e(\phi(y)) = \phi(y')$ and for every constraint $C = (y_1, y_2, R_C)$ we have $(\phi(y_1), \phi(y_2)) \in R_C$.

The apparent size of a forest-CSP instance is the sum of the number of variables, number of trees of $F$, and the number of constraints.

We will show the following result.

**Lemma 3.4.** There exists an algorithm that, given a forest-CSP instance $\mathcal{I}$ of apparent size $s$, in $2^{O(s \log s)}|\mathcal{I}|^{O(1)}$ time computes a satisfying assignment of $\mathcal{I}$ or correctly concludes that $\mathcal{I}$ is unsatisfiable.

To see that Lemma 3.4 implies Theorem 3.2 we translate an auxiliary CSP instance $(X, D, \mathcal{C})$ with $k$ variables into an equivalent forest-CSP instance $(F, D', \mathcal{C}')$. Start with $D = \emptyset$, $\mathcal{C}' = \emptyset$, and a forest $F$ consisting of $k$ components $T_1, T_2, \ldots, T_k$ where $T_i$ is an isolated vertex $x_i \in X$. Define the domain $(D_{T_i}, \leq_{T_i}) \in D'$ of tree $T_i$ as $(D_{T_i}, \leq_{T_i}) := (D, \leq_1) \in D$. Recall that for every constant $C - (x_i(c,1), x_i(c,2), R_C) \in \mathcal{C}$ there is a segment representation, that is, there are $\ell_1, \ell_2 \in \mathbb{N}$, segment reversions $g_1, g_2, \ldots, g_{\ell_1}$ and $g_1', g_2', \ldots, g_{\ell_2}'$, and a downwards-closed relation $R_C'$ such that

$$(a_1, a_2) \in R_C \Rightarrow (g_1^{k_1} \circ g_1^{-1} \circ \ldots \circ g_1^{k_1}(a_1), g_2^{k_2} \circ g_2^{-1} \circ \ldots \circ g_2^{k_2}(a_2)) \in R_C'.$$

For each constraint $C \in \mathcal{C}$ as above, proceed as follows:

1. For each $j = 1, 2$ attach to $x_i(c,j)$ in the tree $T_i(c,j)$ a path of length $k_j$ with vertices $x_i(c,j) = y_{j,1}^k, y_{j,2}^k, \ldots, y_{j,n}^k$, wherein $y_{j,1}, \ldots, y_{j,n}$ are new variables, and label the each edge $y_{j,1}^{k_1-1}y_{j,1}$ with the segment reversion $g_{j,1}^k$.
2. Add a constraint $C' = (y_{1}^{k_1}, y_{2}^{k_2}, R_C')$ to $\mathcal{C}'$.

A direct check shows that a natural extension of a satisfying assignment to the input auxiliary CSP instance $(X, D, \mathcal{C})$ satisfies the resulting forest-CSP instance $(F, D', \mathcal{C}')$ and, in the other direction, a restriction to $\{x_1, x_2, \ldots, x_k\}$ of any satisfying assignment to $(F, D', \mathcal{C}')$ is a satisfying assignment to $(X, D, \mathcal{C})$. Furthermore, if the input auxiliary CSP instance has $k$ variables, $c$ constraints, and $p$ is the sum of the depths of all segment representations, then the apparent size of the resulting forest-CSP instance is $p + 2k + c$. Thus, Theorem 3.2 follows from Lemma 3.4.

The rest of this section is devoted to the proof of Lemma 3.4.
3.1 Fixed-parameter algorithm for forest CSPs

In what follows, to solve a forest-CSP instance means to check its satisfiability and, in case of a satisfiable instance, produce one satisfying assignment. The algorithm for Lemma 3.4 is a branching algorithm that at every recursive call performs a number of preprocessing steps and then branches into a number of subcases. Every recursive call will be performed in polynomial time and will lead to a number of subcalls that is polynomial in s. Every recursive call will be given a forest-CSP instance I and will either solve I directly or produce forest-CSP instances and pass them to recursive subcalls while ensuring that (i) the input instance I is satisfiable if and only if one of the instances passed to the recursive subcalls is satisfiable, and (ii) given a satisfying assignment of an instance passed to a recursive subcall, one can produce a satisfying assignment to I in polynomial time. In that case, we say that the recursive call is correct. In every recursive subcall the apparent size s will decrease by at least one, bounding the depth of the recursion by s. In that case, we say that the recursive call is diminishing. Observe that these two properties guarantee the correctness of the algorithm and the running time bound of Lemma 3.4.

We will often phrase a branching step of a recursive algorithm as guessing a property of a hypothetical satisfying assignment. Formally, at each such step, the algorithm checks all possibilities iteratively.

It will be convenient to assume that every domain (DT, ≤T) equals \{1, 2, ..., |DT|\} with the order ≤T inherited from the integers. (This assumption can be reached by a simple remapping argument and we will maintain it throughout the algorithm.) Thus, henceforth we always use integer order < for the domains.

Let us now focus on a single recursive call. Assume that we are given a forest-CSP instance

\[ I = (F; (DT)_{T \in \text{trees}(F)}, (g_e)_{e \in E(F)}, C) \]

of size s. For two nodes y, y' \in V(F) in the same tree T of F, we denote

\[ g_{y \rightarrow y'} = g_{e_r} \circ g_{e_{r-1}} \circ \ldots \circ g_{e_1}, \]

where e_1, e_2, ..., e_r is the unique path from y to y' in T. Thus, if φ is a satisfying assignment, then φ(y') = g_{y \rightarrow y'}(φ(y)). (And, moreover, φ(y) = g_{y' \rightarrow y}(φ(y')) since each segment reversion g_e satisfies g_e = g_e^{-1}.) In other words, a fixed value of one variable in a tree T fixes the values of all variables in that tree. Thus, there are |DT| possible assignments of all variables of a tree T and we can enumerate them in time \(O(|T| \cdot |DT|)\). We need the following auxiliary operations.

**Forbidding a value.** We define the operation of forbidding value a \in D_{\text{tree}(y)} for variable y \in V(F) as follows. Let T = \text{tree}(y). Intuitively, we would like to delete a from the domain of y and propagate this deletion to all y' \in V(T) and constraints binding variables of T. Formally, we let D'_T = \{1, 2, ..., |DT| - 1\}. For every y' \in V(T), we define α_{y'} : D'_T \rightarrow DT as α_{y'}(b) = b if b < g_{y \rightarrow y'}(a) and α_{y'}(b) = b + 1 if b ≥ g_{y \rightarrow y'}(a). In every constraint C = (y_1, y_2, RC) and j \in \{1, 2\}, if y_j \in V(T), then we replace RC with R'_C defined as follows,

\[
R'_C = \{(x_1, x_2) \in D'_T \times D_{\text{tree}(y_2)} \mid (α_{y_1}(x_1), x_2) \in R_C\} \quad \text{if } j = 1,
\]

\[
R'_C = \{(x_1, x_2) \in D_{\text{tree}(y_1)} \times D'_T \mid (x_1, α_{y_1}(x_2)) \in R_C\} \quad \text{if } j = 2.
\]

(Note that y_1 and y_2 are not necessarily in different trees.) Observe that each domain remains of the form \{0, 1, ..., ℓ\} for some ℓ \in \mathbb{N}. It is straightforward to verify that R'_C is downwards-closed as R_C is downwards-closed. Furthermore, a direct check shows that:

1. If φ is a satisfying assignment to the original instance such that φ(y) ≠ a, then φ(y') ≠ g_{y \rightarrow y'}(a) for every y' \in V(T). Moreover, the assignment φ' defined as φ'(y') = α_{y'}^{-1}(φ(y')) for every y' \in V(T) and φ'(y') = φ(y') for every y' \in V(F) \setminus V(T) is a satisfying assignment to the resulting instance.
2. If φ' is a satisfying assignment to the resulting instance, then φ defined as φ(y') = α_{y'}(φ'(y')) for every y' \in V(T), and φ(y') = φ'(y') for every y' \in V(F) \setminus V(T) is a satisfying assignment to the original instance.
Restricting the domain $D_T$ of a variable $y \in V(T)$ to $A \subseteq D_T$ means forbidding all values of $D_T \setminus A$ for $y$.

We now describe the steps performed in the recursive call and argue in parallel that the recursive call is correct and diminishing.

**Preprocessing steps.** We perform the following preprocessing steps exhaustively.

1. If there are either no variables (hence a trivial empty satisfying assignment) or a variable with an empty domain (hence an obvious negative answer), solve the instance directly.

   Thus, henceforth we assume $V(F) \neq \emptyset$ and that every domain is nonempty.

2. For every constraint $C$ that binds two variables from the same tree $T$, we iterate over all $|D_T|$ possible assignments of all variables in $T$ and forbid those that do not satisfy $C$. (Recall that fixing the value of one variable of a tree fixes the values of all other variables of that tree.) Finally, we delete $C$.

   Thus, henceforth we assume that every constraint binds variables from two distinct trees of $F$.

3. For every constraint $C$, for both variables $y_j$, $j = 1, 2$, that are bound by $C$, and for every $a \in D_{\text{tree}(y_j)}$, if there is no $b \in D_{\text{tree}(y_{3-j})}$ such that $(a, b)$ satisfies $C$, we forbid $a$ for the variable $y_j$.

   Thus, henceforth we assume that for every constraint $C$, every variable it binds, and every possible value $a$ of this variable, there is at least one value of the other variable bound by $C$ that together with $a$ satisfies $C$.

Clearly, the above preprocessing steps can be performed exhaustively in polynomial time and they do not increase the apparent size of the instance.

We next perform three branching steps. Ultimately, in each of the subcases we consider we will make a recursive call. However, the branching steps 1 and 2 both hand one subcase down for treatment in the later branching steps.

For every $T \in \text{trees}(F)$, pick arbitrarily some node $x_T \in V(T)$. Assume that $I$ is satisfiable and let $\phi$ be a satisfying assignment that is minimal in the following sense. For every $T \in \text{trees}(F)$, we require that either $\phi(x_T) = 1$ or if we replace the value $\phi(x_T)$ with $\phi(x_T) - 1$ and the value $\phi(y)$ with $g_{x_T\rightarrow y}(\phi(x_T) - 1)$ for every $y \in V(T)$, we violate some constraint. Note that if $I$ is satisfiable then such an assignment exists, because each domain $D_T$ has the form $\{1, 2, \ldots, |D_T|\}$ and thus $\phi(x_T) - 1, g_{x_T\rightarrow y}(\phi(x_T) - 1) \in D_T$.

**First branching step.** We branch into $1 + |\text{trees}(T)| \leq s + 1$ subcases, guessing whether there exists a tree $T$ such that the variable $x_T$ satisfies $\phi(x_T) = 1$ and which tree it is precisely. If we have guessed that no such tree exists, we proceed to the next steps of the algorithm with the assumption that $\phi(x_T) > 1$ for every $T \in \text{trees}(F)$. The other subcases are labelled by the trees of $F$. In the subcase for $T \in \text{trees}(F)$, we guess that $\phi(x_T) = 1$. For every constraint $C = (y_1, y_2, R_C)$ that binds $y_j \in V(T)$ with another variable $y_{3-j} \notin V(T)$, we restrict the domain $D_{\text{tree}(y_{3-j})}$ of $y_{3-j}$ to only values $b$ such that $(g_{x_T\rightarrow y_1}(1), b) \in R_C$. Finally, we delete the tree $T$ and all constraints binding variables of $V(T)$, and invoke a recursive call on the resulting instance.

To see that this step is diminishing, note that, due to the deletion of $T$, the apparent size in the recursive call is reduced by at least one. For correctness, clearly, if $\phi(x_T) = 1$, then the resulting instance is satisfiable and any satisfying assignment to the resulting instance can be extended to a satisfying assignment of $I$ by assigning $g_{x_T\rightarrow y}(1)$ to $y$ for every $y \in V(T)$.

**Second branching step.** We guess whether there exists an edge $yy' \in E(F)$ such that $\phi(y)$ is an endpoint of a segment of $g_{yy'}$. If we have guessed that no such edge $yy'$ exists, we proceed to the next steps of the algorithm. Otherwise, we guess $yy' \in E(F)$, one endpoint $y$, and whether $\phi(y)$ is the left or the right endpoint of a segment of $g_{yy'}$, leading to at most $4|E(F)| \leq 4s$ subcases. (Note that $|E(F)| \leq |V(F)| \leq s$.) We restrict the domain $D_{\text{tree}(y)}$ of $y$ to only those values $a$ such that $a$ is the left/right (according to the
guess) endpoint of a segment of $g_{yy'}$. Observe that now $g_{yy'}$ is an identity, as each of its segment has been reduced to a singleton. Consequently, we do not change the set of satisfying assignments if we contract the edge $yy'$ in the tree $\text{tree}(y)$ and, for every constraint binding $y$ or $y'$, modify $C$ to bind instead the image of the contraction of the edge $yy'$. This decreases $s$ by one and we pass the resulting instance to a recursive subcall.

**Third branching step.** Hence, we proceed to the last branching step with the case where no edge $yy'$ as in branching step 2 exists. Recall that also from the first branching step we can assume that $\phi(x_T) > 1$ for every $T \in \text{trees}(F)$. Pick an arbitrary tree $T \in \text{trees}(F)$. Using the minimality of $\phi$, we now guess which constraint $\Gamma = (y_1, y_2, R_T)$ is violated if we replace $\phi(x_T)$ with $\phi(x_T) - 1$ and $\phi(y)$ with $g_{x_T \to y}(\phi(x_T) - 1)$ for every $y \in V(T)$. By symmetry, assume $y_1 \in V(T)$. Since, due to preprocessing, every constraint binds variables of two distinct trees, $y_2 \notin V(T)$. Let $S = \text{tree}(y_2)$. Note that we have at most $s$ subcases in this branching step.

We now aim to show that assigning a value to $y_1$ fixes the value of $y_2$ via constraint $\Gamma$. Consequently, we will be able to remove $\Gamma$ and merge the trees $S$ and $T$, resulting in a smaller forest-CSP instance, which we can solve recursively.

Recall that for every $a \in D_S$ there exists at least one $b \in D_T$ with $(b, a) \in R_T$, by preprocessing step 3. Since $R_T$ is a downwards-closed relation, there exists a nonincreasing function $f' : D_S \to D_T$ such that

$$R_T = \{(b, a) \in D_T \times D_S \mid b \leq f'(a)\}.$$ 

The crucial observation is the following.

**Claim 3.5.** Assume that $\phi$ exists and all guesses in the current recursive call have been made correctly. Then, $\phi(y_1) = f'(\phi(y_2))$.

**Proof.** Since we made a correct guess at the second branching step, for every edge $yy'$ on the path in $T$ from $x_T$ to $y_1$ (with $y'$ closer than $y$ to $x_T$), the value $\phi(y) = g_{x_T \to y}(\phi(x_T))$ is not an endpoint of $g_{yy'}$. Inductively from $x_T$ to $y_1$, we infer that for every $y$ on the path from $y_1$ to $x_T$ we have that $\phi(y) = g_{x_T \to y}(\phi(x_T))$ and $g_{x_T \to y}(\phi(x_T) - 1)$ are two consecutive integers. In particular, $\phi(y_1)$ and $g_{x_T \to y_1}(\phi(x_T) - 1)$ are two consecutive integers.

By choice of $\Gamma$, we have $(g_{x_T \to y_1}(\phi(x_T) - 1), \phi(y_2)) \notin R_T$ but $(\phi(y_1), \phi(y_2)) \in R_T$. Since $R_T$ is downwards-closed, this is only possible if $g_{x_T \to y_1}(\phi(x_T) - 1) = \phi(y_1) + 1$ and hence $\phi(y_1) = f'(\phi(y_2))$. This concludes the proof of the claim.

Claim 3.5 implies that by fixing an assignment of the tree $S$, we induce an assignment of $T$ via the function $f'$. We would like to merge the two trees $S$ and $T$ via an edge $yy'$, labelled with $f'$. However, $f'$ is not a segment reversion, but a nonincreasing function. Thus, we need to perform some work to get back to a forest-CSP instance representation. For this, we will leverage Lemma 2.7

Let $g^\circ$ be a segment reversion with one segment, reversing the whole $D_S$. Let $f'' = f' \circ g^\circ$, that is, $f'' : D_S \to D_T$ and $f'' \circ g^\circ = f'$. Observe that since $f'$ is nonincreasing, $f''$ is nondecreasing.

We perform the following operation on $T$ that will result in defining segment reversions $g'_e$ of $D_T$ for every $e \in E(T)$ and nondecreasing functions $f_y : D_S \to D_T$ for every $y \in V(T)$ as follows. We temporarily root $T$ at $y_1$. We initiate $f_{y_1} = f''$. Then, in a top-to-bottom manner, for every edge $yy'$ between a node $y$ and its parent $y'$ such that $f_{y'}$ is already defined, we invoke Lemma 2.7 to $f_{y'}$ and the segment reversion $g_{yy'}$, obtaining a segment reversion $g'_{yy'}$ of $D_S$ and a nondecreasing function $f_y : D_S \to D_T$ such that

$$g_{yy'} \circ f_y = f_y \circ g'_{yy'}.$$  

We merge the trees $S$ and $T$ into one tree $T'$ by adding an edge $yy_2$ and define $g'_{yy_2} = g^\circ$. We set $D_{T'} = D_S$; observe that all $g'_e$ for $e \in E(T)$ as well as $g'_{yy_2}$ are segment reversions of $D_{T'}$. Let $F'$ be the resulting forest. For every $e \in E(F) \setminus E(T)$, we define $g'_e = g_e$. Similarly as we defined $g_{yy'}$, we define $g'_{yy'}$ for every two vertices $y, y'$ of the same tree of $F'$ as $g'_{y_1} \circ g'_{y_2} \circ \cdots \circ g'_{y'}$ where $e_1, e_2, \ldots, e_r$ are the edges on the path from $y$ to $y'$ in $F'$. Note that $g'_{yy'} = g_{yy'}$ when $y, y' \notin V(T')$ or $y, y' \in V(S)$.  

10
We now define a modified set of constraints $\mathcal{C}'$ as follows. Every constraint $C \in \mathcal{C}$ that does not bind any variable of $T$ we insert into $\mathcal{C}'$ without modifications. For every constraint $C \in \mathcal{C}$ that binds a variable of $T$, we proceed as follows. By symmetry, assume that $C = (z_1, z_2, R_C)$ with $z_1 \in V(T)$ and $z_2 \notin V(T)$. Recall that $R_C \subseteq D_T \times D_{\text{tree}(z_2)}$ and $f_{z_1} : D_S \to D_T$. We apply Lemma 2.8 to $R_C$ and $f_{z_1}$, obtaining a downwards-closed relation $R'_C \subseteq D_S \times D_{\text{tree}(z_2)}$ such that

$$(a, b) \in R'_C \iff (f_{z_1}(a), b) \in R_C.$$  

We insert $C' := (z_1, z_2, R'_C)$ into $\mathcal{C}'$.

Let $\mathcal{T}' = (F', (D_T)_{T \in \text{trees}(F')}, (g'_C)_{C \in \mathcal{C}'})$ be the resulting forest-CSP instance. Note that $|V(F')| \leq |V(F)|$, $|\mathcal{C}'| \leq |\mathcal{C}|$, while $|\text{trees}(F')| < |\text{trees}(F)|$. Thus, the apparent size of $\mathcal{T}'$ is smaller than the apparent size of $\mathcal{T}$. We pass $\mathcal{T}'$ to a recursive subcall.

To complete the proof of Lemma 3.4, it remains to show correctness of branching step 3. This is done in the next two claims.

Claim 3.6. Let $\zeta'$ be a satisfying assignment to $\mathcal{T}'$. Define an assignment $\zeta$ to $\mathcal{T}$ as follows. For every $y \in V(F) \setminus V(T)$, set $\zeta(y) = \zeta'(y)$. For every $y \in V(T)$, set $\zeta(y) = f_y(\zeta'(y))$. Then $\zeta$ is a satisfying assignment to $\mathcal{T}$.

Proof. To see that $\zeta$ is an assignment, that is, maps each variable into its domain, since every function $f_y$ for $y \in V(T)$ has domain $D_S = D_T$ and codomain $D_T$, every $y \in V(T)$ satisfies $\zeta(y) \in D_T$.

To see that $\zeta$ is a satisfying assignment, consider first the condition on the forest edges. Pick $e = yy' \in E(F)$. If $e \notin E(T)$, then $\zeta(y) = \zeta'(y)$, $\zeta(y') = \zeta'(y')$, $g_e = g'_e$, and obviously $\zeta(y') = g'_e(\zeta(y))$. Otherwise, assume without loss of generality that $y'$ is closer than $y$ to $y_1$ in $T$. Then (2) ensures that

$$g_{yy'}(\zeta'(y')) = g_{yy'}(f_y'(\zeta'(y'))) = f_y(g'_{yy'})(\zeta'(y')) = f_y(\zeta'(y')) = \zeta(y)$$

as desired.

Now pick a constraint $C \in \mathcal{C}$ and let us show that $\zeta$ satisfies $C$. If $C$ does not bind a variable of $T$, then $C' \in \mathcal{C}'$ and $\zeta$ and $\zeta'$ agree on the variables bound by $C$, hence $\zeta$ satisfies $C$. Otherwise, without loss of generality, $C = (z_1, z_2, R_C)$ with $z_1 \in V(T)$ and there is the corresponding constraint $C' = (z_1, z_2, R'_C)$ in $\mathcal{C}'$ as defined above. Since $\zeta'$ satisfies $C'$, we have $(\zeta'(z_1), \zeta'(z_2)) \in R'_C$. By the definition of $R'_C$, this is equivalent to $(f_{z_1}(\zeta'(z_1)), \zeta'(z_2)) \in R_C$. Since $\zeta(z_1) = f_{z_1}(\zeta'(z_1))$ (as $z_1 \in V(T)$) and $\zeta(z_2) = \zeta'(z_2)$, this is equivalent to $(\zeta(z_1), \zeta(z_2)) \in R_C$. Hence, $\zeta$ satisfies the constraint $C$. This finishes the proof of the claim.

Claim 3.7. Let $\zeta$ be a satisfying assignment to $\mathcal{T}$ that additionally satisfies $\zeta(y_1) = f'(\zeta(y_2))$. Define an assignment $\zeta'$ to $\mathcal{T}'$ as follows. For every $y \in V(F) \setminus V(T)$, set $\zeta'(y) = \zeta(y)$. For every $y \in V(T)$, set $\zeta'(y) = g'_{y_2\rightarrow y}(\zeta(y_2))$. Then $\zeta'$ is a satisfying assignment to $\mathcal{T}'$.

Proof. To see that $\zeta'$ is indeed an assignment, it is immediate from the definition of $\mathcal{T}'$ that for every tree $A$ of $F'$ and $y \in V(A)$ we have $\zeta'(y) \in D_A$. To see that $\zeta'$ is a satisfying assignment, by definition, for every $e = yy' \in E(F')$ we have $\zeta'(y') = g'_e(\zeta'(y))$. Also, obviously $\zeta'$ satisfies all constraints of $\mathcal{C}'$ that come unmodified from a constraint of $\mathcal{C}$ that does not bind a variable of $V(T)$. It remains to show that the remaining constraints are satisfied.

Consider a constraint $C' = (z_1, z_2, R'_C) \in \mathcal{C}'$ that comes from a constraint $C = (z_1, z_2, R_C) \in \mathcal{C}$ binding a variable of $V(T)$. Without loss of generality, $z_1 \in V(T)$ and $z_2 \notin V(T)$. By composing (2) over all edges on the path from $z_1$ to $y_1$ in $T$ we obtain that

$$g_{y_1\rightarrow z_1} \circ f'' = f_{z_1} \circ g'_{y_1\rightarrow z_1}.$$  

By composing the above with $g^o$ on the right and using $f'' = f' \circ g^o$ (hence $f'' \circ g^o = f'$) and $g^o = g'_{y_1\rightarrow y_2}$, we obtain that

$$g_{y_1\rightarrow z_1} \circ f' = f_{z_1} \circ g'_{y_2\rightarrow z_1}. \quad (3)$$
By the definition of $R'_C$, we have that $(\zeta'(z_1), \zeta'(z_2)) \in R'_C$ is equivalent to
\[(f_{z_1}(\zeta'(z_1)), \zeta'(z_2)) \in R_C.\]

By the definition of $\zeta'$, this is equivalent to
\[(f_{z_1} \circ g'_{y_2 \to z_2}(\zeta(y_2)), \zeta(z_2)) \in R_C.\]

By (3), this is equivalent to
\[(g_{y_1 \to z_1} \circ f'(\zeta(y_2)), \zeta(z_2)) \in R_C.\]

Since $f'(\zeta(y_2)) = \zeta(y_1)$, this is equivalent to
\[(g_{y_1 \to z_1}(\zeta(y_1)), \zeta(z_2)) \in R_C.\]

By the definition of $g_{y_1 \to z_1}$, this is in turn equivalent to
\[(\zeta(z_1), \zeta(z_2)) \in R_C,\]

which follows as $\zeta$ satisfies $C$. This finishes the proof of the claim.

Claims 3.6 and 3.7 show the correctness of the third branching step, concluding the proof of Lemma 3.4 and of Theorem 3.2.

## 4 From Optimal Discretization to the auxiliary CSP

To prove Theorem 1.1 we give an algorithm that constructs a branching tree. At each branch, the algorithm tries a limited number of options for some property of the solution. At the leaves it will then assume that the chosen options are correct and reduce the resulting restricted instance of OPTIMAL DISCRETIZATION to the auxiliary CSP from Section 3. We first give basic notation for the building blocks of the solution in Section 4.1. The branching tree is described in Section 4.2. The reduction to the auxiliary CSP is given in Sections 4.3 and 4.5 to 4.7. Throughout the description of the algorithm, we directly argue that it satisfies the running time bound and that it is sound, meaning that, if there is a solution, then a solution will be found in some branch of the branching tree. We argue in the end, in Sections 4.8 and 4.9, that the algorithm is complete, that is, if it does not return that the input is a no-instance, then the returned object is a solution.

### 4.1 Approximate solution and cells

Let $(W_1, W_2, k)$ be an input to the decision version of OPTIMAL DISCRETIZATION. We assume that $W_1 \cap W_2 = \emptyset$, as otherwise there is no solution.

Using a known factor-2 approximation algorithm [4], we compute in polynomial time a separation $(X_0, Y_0)$. If $|X_0| + |Y_0| > 2k$, we report that the input instance is a no-instance. Otherwise, we proceed further as follows.

**Discretization.** Let $n = |W_1| + |W_2|$. By simple discretization and rescaling, we can assume that

- every point in $W_1 \cup W_2$ has both coordinates being positive integers from $[3n]$ and divisible by 3,
- the sought solution $(X, Y)$ consists of integers from $[3n]$ that are equal to 2 modulo 3,
- every element of $X_0 \cup Y_0$ is an integer from $[3n]$ that is equal to 1 modulo 3.

Furthermore, we add 1 and $3n + 1$ to both $X_0$ and $Y_0$ (if not already present). Thus, $|X_0| + |Y_0| \leq 2k + 4$, $X_0, Y_0 \subseteq \{3i + 1 \mid i \in \{0, 1, \ldots, n\}\}$ and for every $(x, y) \in W_1 \cup W_2$ we have that $x$ is between the minimum and maximum element of $X_0$ and $y$ is between the minimum and maximum element of $Y_0$. We henceforth refer to the properties obtained in this paragraph as the discretization properties.
Total orders \( \leq_{x} \leq_{y} \). We will use two total orders on points of \( W_1 \cup W_2 \):

- \((x, y) \leq_{x} (x', y')\) if \( x < x' \) or both \( x = x' \) and \( y \leq y' \);
- \((x, y) \leq_{y} (x', y')\) if \( y < y' \) or both \( y = y' \) and \( x \leq x' \).

For a set \( W \subseteq W_1 \cup W_2 \), the topmost point is the \( \leq_{y} \)-maximum one, the bottommost is the \( \leq_{y} \)-minimum, the leftmost is the \( \leq_{x} \)-minimum, and the rightmost is the \( \leq_{x} \)-maximum one. Finally, an extremal point in \( W \) is the topmost, bottommost, leftmost, or the rightmost point in \( W \); there are at most four extremal points in a set \( W \).

Assume that the input instance is a yes-instance and let \((X, Y)\) be a sought solution: a separation for \((W_1, W_2)\) with \(|X| + |Y| \leq k\) and \(X, Y \subseteq \{3i - 1 \mid i \in [n]\}\).

Cells. For two consecutive elements \( x_1, x_2 \) of \( X_0 \cup X \) and two consecutive elements \( y_1, y_2 \) of \( Y_0 \cup Y \), define the set \( \text{cell}(x_1, y_1) \) := \{\(x_1 + 1, x_1 + 2, \ldots, x_2 - 1\} \times \{y_1 + 1, y_1 + 2, \ldots, y_2 - 1\}\). Each such set is called a cell. Note that since we require \( x_1, x_2 \) to be consecutive elements of \( X_0 \cup X \) and similarly \( y_1, y_2 \) to be consecutive elements of \( Y_0 \cup Y \), the pair \((x_1, y_1)\) determines the corresponding cell uniquely. The points in the cell \( \text{cell}(x_1, y_1) \) are the points in the set \( \text{cell}(x_1, y_1) \cap (W_1 \cup W_2) \).

Similarly, for two consecutive elements \( x_1, x_2 \) of \( X_0 \) and two consecutive elements \( y_1, y_2 \) of \( Y_0 \), an apx-supercell is the set \( \text{apxcell}(x_1, y_1) := \{x_1 + 1, x_1 + 2, \ldots, x_2 - 1\} \times \{y_1 + 1, y_1 + 2, \ldots, y_2 - 1\}\) and the points in this cell are \( \text{apxcell}(x_1, y_1) \cap (W_1 \cup W_2) \). Also, for two consecutive elements \( x_1, x_2 \) of \( X \cup \{1, 3n + 1\} \) and two consecutive elements \( y_1, y_2 \) of \( Y \cup \{1, 3n + 1\} \), an opt-supercell is the set \( \text{optcell}(x_1, y_1) := \{x_1 + 1, x_1 + 2, \ldots, x_2 - 1\} \times \{y_1 + 1, y_1 + 2, \ldots, y_2 - 1\}\) and the points in this cell are \( \text{optcell}(x_1, y_1) \cap (W_1 \cup W_2) \).

Clearly, every apx-supercell or opt-supercell contains a number of cells and each cell is contained in exactly one apx-supercell and exactly one opt-supercell. Note that, since \((X_0, Y_0)\) and \((X, Y)\) are separations, all points in one cell, in one apx-supercell, and in one opt-supercell are either from \( W_1 \) or from \( W_2 \), or the (super)cell contains no points.

Furthermore, observe that there are \(O(k^2)\) cells, apx-supercells, and opt-supercells.

We will also need the following general notation. For two elements \( x_1, x_2 \in X_0 \cup X \) with \( x_1 < x_2 \) and two elements \( y_1, y_2 \in Y_0 \cup Y \) with \( y_1 < y_2 \), we denote the union of all cells \( \text{cell}(x, y) \) that are between \( x_1 \) and \( x_2 \) and between \( y_1 \) and \( y_2 \), that is, that satisfy \( x_1 \leq x < x_2 \) and \( y_1 \leq y < y_2 \), that is, that satisfy \( x_1 \leq x < x_2 \) and \( y_1 \leq y < y_2 \).

4.2 Branching steps

In the algorithm we first perform a number of branching steps. Every step is described in the “intuitive” language of guessing a property of the solution. Formally, at every step we are interested in some property of the solution with some (bounded as a function of \( k \)) number of options and we consider all possible options iteratively. While considering one of these options, we are interested in finding some solution to the input instance in case \((X, Y)\) satisfies the considered option. If the solution satisfies the currently considered option, we also say that the corresponding guess is correct.

Branching step A: separating elements of the solution. We guess whether there exists an apx-supercell and an extremal point \((x, y)\) in this cell such that (see Figure 2)

1. for some \( x_1, x_2 \in X \), \( x \) is between \( x_1 \) and \( x_2 \) while no element of \( X_0 \) is between \( x_1 \) and \( x_2 \), or
2. for some \( y_1, y_2 \in Y \), \( y \) is between \( y_1 \) and \( y_2 \) while no element of \( Y_0 \) is between \( y_1 \) and \( y_2 \).

If we have guessed that this is the case, then we guess \((x, y)\) and, in the first case, we add \( x + 1 \) to \( X_0 \), and in the second case we add \( y + 1 \) to \( Y_0 \), and recursively invoke the same branching step. If we have guessed that no such apx-cell and an extremal point exist, then we proceed to the next steps of the algorithm.

As \(|X| + |Y| \leq k\), the above branching step can be correctly guessed and executed at most \( k - 1 \) times. Hence, we limit the depth of the branching tree by \( k - 1 \): at a recursive call at depth \( k - 1 \) we only consider the case where no such extremal point \((x, y)\) exists.

At every step of the branching process, there are \(O(k^2)\) apx-supercells to choose, at most four extremal points in every cell, and two options whether the \( x \)-coordinate of the extremal point separates two elements.
of $X$ or the $y$-coordinate of the extremal point separates two elements of $Y$. Thus, the whole branching process in this step generates $2^{O(k \log k)}$ cases to consider in the remainder of the algorithm, where we can assume that no such extremal point $(x, y)$ in any $\text{apx}$-supercell exists. Note that the branching does not violate the discretization properties of the elements of $W_1, W_2, X, Y, X_0$, and $Y_0$ and keeps $|X_0| + |Y_0| \leq (2k + 4) + (k - 1) = 3k + 3$.

**Branching step B: layout of the solution with regard to the approximate one.** For every two consecutive elements $x_1, x_2 \in X_0$, we guess the number of elements $x \in X$ that are between $x_1$ and $x_2$, and similarly for every two consecutive elements $y_1, y_2 \in Y_0$, we guess the number of elements $y \in Y$ that are between $y_1$ and $y_2$. Recall that $X_0 \cap X = \emptyset$, $Y_0 \cap Y = \emptyset$, and that $1, 3n + 1 \in X_0 \cap Y_0$, so every element of $X$ and $Y$ is between two consecutive elements of $X_0$ or $Y_0$, respectively. Furthermore, since $|X| + |Y| \leq k$ and $|X_0| + |Y_0| \leq 3k + 3$, the above branching leads to $2^{O(k)}$ subcases.

**The notions of abstract lines, cells, and their corresponding mappings $\zeta$.** Observe that if we have guessed correctly in Branching Step B, we know $|X \cup X_0|$ and, if we order $X \cup X_0$ in the increasing order, we know which elements of $X \cup X_0$ belong to $X$ and which to $X_0$; a similar claim holds for $Y \cup Y_0$. We “only” do not know the exact values of the elements of $X$ and $Y$, but we have a rough picture of the layout of the cells. We abstract this information as follows.

We create a totally ordered set $(X_{\text{lin}}, \prec)$ of $|X \cup X_0|$ elements which we will later refer to as *vertical lines*. Let $X_{\text{lin}}^{\text{apx}} = (\zeta_X^X)^{-1}(X_0)$ be the lines corresponding to the elements of $X_0$ and let $X_{\text{lin}}^{\text{opt}} = X_{\text{lin}} \setminus X_{\text{lin}}^{\text{apx}}$. Denote $\zeta_X^{\text{apx}} = \zeta_X^X|_{X_{\text{lin}}^{\text{apx}}}$ and $\zeta_X^{\text{opt}} = \zeta_X^X|_{X_{\text{lin}}^{\text{opt}}}$. Similarly, we define a totally ordered set $(Y_{\text{lin}}, \prec)$ of $|Y \cup Y_0|$ *horizontal lines*, sets $Y_{\text{lin}}^{\text{apx}}, Y_{\text{lin}}^{\text{opt}} \subseteq Y_{\text{lin}}$, and functions $\zeta_Y^{\text{apx}}, \zeta_Y^{\text{opt}}$. Finally, we define $\zeta_{\text{apx}} = \zeta_X^{\text{apx}} \cup \zeta_Y^{\text{apx}}$.

Observe that while $\zeta_X^X, \zeta_X^{\text{opt}}, \zeta_Y^Y$, and $\zeta_Y^{\text{opt}}$ depend on the (unknown to the algorithm) solution $(X, Y)$, the sets $X_{\text{lin}}^{\text{apx}}, Y_{\text{lin}}^{\text{apx}}, X_{\text{lin}}^{\text{opt}}, Y_{\text{lin}}^{\text{opt}}$, and functions $\zeta_X^{\text{apx}}, \zeta_Y^{\text{apx}}$, and $\zeta_{\text{apx}}$ do not depend on $(X, Y)$ and can be computed by the algorithm. This is why we avoid the subscript $X$ or $Y$ in $\zeta_X^{\text{apx}}, \zeta_Y^{\text{apx}}$, and $\zeta_{\text{apx}}$.

---

4Respecting the orders means that for each $x, y \in X_{\text{lin}}$ we have that, if $x \leq y$, then $\zeta_X^X(x) \leq \zeta_X^X(y)$.

5Let $f : A \to B$ and $C \subseteq A$. Then $f|_C$ is the function resulting from $f$ when removing $A \setminus C$ from the domain of $f$.  

---

Figure 2: Branching Step A. Adding to $X_{\text{apx}}^\text{lin}$ (solid) an extra horizontal line (gray) just above the rightmost red point (circled) separates some horizontal lines from the solution (dashed) that were not separated before.
Our goal can be stated as follows: we want to extend $\zeta^x,\text{apx}$ and $\zeta^y,\text{apx}$ to increasing functions $\zeta^x : X_{\text{in}} \to \mathbb{N}$ and $\zeta^y : Y_{\text{in}} \to \mathbb{N}$ such that $\{\zeta^x(\ell) \mid \ell \in X_{\text{opt}}\}$ and $\{\zeta^y(\ell) \mid \ell \in Y_{\text{opt}}\}$ is a separation.

Recall that the notions of cells, apx-supercells, and opt-supercells, as well as the notion area, have been defined with regard to the solution $(X,Y)$, but we can also define them with regard to lines $X_{\text{in}}$ and $Y_{\text{in}}$. That is, for a cell $\text{cell}(x_1,y_1)$, its corresponding abstract cell is $\text{cell}((\zeta^x_x)^{-1}(x_1), (\zeta^y_y)^{-1}(y_1))$. Let $X_{\text{in}}^-$ be the set $X_{\text{in}}$ without the maximum element and $Y_{\text{in}}^-$ be the set $Y_{\text{in}}$ without the maximum element. Then we denote the set of abstract cells by $\text{Cells} = \{\text{cell}(\ell_x, \ell_y) \mid \ell_x \in X_{\text{in}}^- \land \ell_y \in Y_{\text{in}}^-\}$. Let $\text{cell}(\ell_x, \ell_y)$ be in $\text{Cells}$ where $\ell'_x$ is the successor of $\ell_x$ in $X_{\text{in}},$ $<$. Then we say that $\ell_x$ is the left side, $\ell_y$ is the bottom side, $\ell'_x$ is the right side, and $\ell'_y$ is the top side of $\text{cell}(\ell_x, \ell_y)$.

Similarly we define abstract apx-supercells and abstract opt-supercells, and the notion area($p_1,p_2,\ell_1,\ell_2$) for $p_1,p_2 \in X_{\text{in}},$ $p_1 < p_2$, $\ell_1,\ell_2 \in Y_{\text{in}},$ $\ell_1 < \ell_2$. If it does not cause confusion, in what follows we implicitly identify the abstract cell $\text{cell}(\ell_x, \ell_y)$ with its corresponding cell $\text{cell}(\zeta^x_x(\ell_x), \zeta^y_y(\ell_y))$ and similarly for apx-supercells and opt-supercells. Note that for apx-supercells the distinction between apx-supercells and abstract apx-supercells is only in notation as the functions $\zeta^x,\text{apx}$ and $\zeta^y,\text{apx}$ are known to the algorithm.

**Branching step C: contents of the cells and associated mapping $\delta$.** For every abstract cell $\text{cell}(\ell_x, \ell_y)$, we guess whether the cell $\text{cell}(\zeta^x_x(\ell_x), \zeta^y_y(\ell_y))$ contains at least one point of $W_1 \cup W_2$. Since there are $O(k^2)$ cells and two options for each cell, this leads to $2^{O(k^2)}$ subcases. Note that if $\text{cell}(\zeta^x_x(\ell_x), \zeta^y_y(\ell_y))$ is guessed to contain some points of $W_1 \cup W_2$, we know whether these points are from $W_1$ or from $W_2$: They are from the same set as the points contained in the apx-supercell containing $\text{cell}(\ell_x, \ell_y)$. (If the corresponding apx-supercell does not contain any points of $W_1 \cup W_2$, we discard the cases when $\text{cell}(\zeta^x_x(\ell_x), \zeta^y_y(\ell_y))$ is guessed to contain points of $W_1 \cup W_2$.) Thus, in fact every cell $\text{cell}(\ell_x, \ell_y)$ can be of one of three types: either containing some points of $W_1$ (type 1), containing some points of $W_2$ (type 2), or not containing any points of $W_1 \cup W_2$ at all (type 0). Let $\delta : \text{Cells} \to \{0,1,2\}$ be the guessed function assigning to every cell its type.

Upon this step, we discard a guess if there are two cells $\text{cell}(\ell_x, \ell_y)$ and $\text{cell}(\ell'_x, \ell'_y)$ such that we have guessed one to contain some points of $W_1$ and the other to contain some points of $W_2$ that are contained in the same opt-supercell, as such a situation would contradict the fact that $(X,Y)$ is a separation. Consequently, we can extend the function $\delta$ to the set of opt-supercells, indicating for every opt-supercell whether at least one cell contains a point of $W_1$, a point of $W_2$, or whether the entire opt-supercell is empty.

For notational convenience, we also extend the function $\delta$ to apx-supercells in the natural manner. Here, we also discard the current guess if there is an apx-supercell that contains some points of $W_1 \cup W_2$, but all abstract cells inside this apx-supercell are of type 0.

**Branching step D: cells of the extremal points and associated mapping $\phi$.** We would like now to guess a function $\phi : W_1 \cup W_2 \to \text{Cells}$ that, for every point $(x,y) \in W_1 \cup W_2$ that is extremal in its cell, assigns to $(x,y)$ the abstract cell $\text{cell}(\ell_x, \ell_y)$ such that $\text{cell}(\zeta^x_x(\ell_x), \zeta^y_y(\ell_y))$ contains $(x,y)$. (And we have no requirement on $\phi$ for points that are not extremal in their cell.)

Consider first a random procedure that for every $(x,y) \in W_1 \cup W_2$ samples $\phi(x,y) \in \text{Cells}$ uniformly at random. Since there are $O(k^2)$ cells and at most four extremal points in one cell, the success probability of this procedure is $2^{-O(k^2 \log k)}$.

This random process can be derandomized in a standard manner using the notion of splitters [1] (see e.g. Cygan et al. [4] for an exposition). For integers $n$, $a$, and $b$, a $(n,a,b)$-splitter is a family $F$ of functions from $[n]$ to $[b]$ such that for every $A \subseteq [n]$ of size at most $a$ there exists $f \in F$ that is injective on $A$. Given integers $n$ and $r$, one can construct in polynomial in $n$ and $r$ time an $(n,r,r^2)$-splitter of size $r^{O(1)} \log n$ [1]. We set $n = |W_1 \cup W_2|$ and $r = 4|\text{Cells}| = O(k^2)$ and construct an $(n,r,r^2)$-splitter $F_1$ where we treat every function $f_1 \in F_1$ as a function with domain $W_1 \cup W_2$. We construct a set $F_2$ of functions from $r^2$ to $\text{Cells}$ as follows: for every set $A \subseteq [r^2]$ of size at most $r = 4|\text{Cells}|$ and every function $f'_2$ from $A$ to $\text{Cells}$, we extend $f'_2$ to a function $f_2 : [r^2] \to \text{Cells}$ arbitrarily (e.g., by assigning to every element of $[r^2] \setminus A$ one fixed element of $\text{Cells}$) and insert $f_2$ into $F_2$. Finally, we define $F = \{f_2 \circ f_1 \mid (f_1, f_2) \in F_1 \times F_2\}$.

Note that $|F| = 2^{O(k^2 \log k)} \log n$ as $F_1$ is of size $k^{O(1)} \log n$ while $F_2$ is of size $2^{O(k^2 \log k)}$ as there are $2^{O(k^2 \log k)}$ choices of the set $A$ and $2^{O(k^2 \log k)}$ choices for the function $f'_2$ from $A$ to $\text{Cells}$. 


Branching Step E: order of the extremal points. For every two abstract cells cell, cell′ ∈ Cells and every two directions Δ, Δ′ ∈ {top, bottom, left, right}, we guess how the Δ-most point in cell (the extremal point in cell in the direction Δ) and the Δ′-most point in cell′ relate in the orders ≤_x and ≤_y. Since ≤_x and ≤_y are total orders and there are O(k^2) extremal points in total, this branching step leads to 2^Ω(k^2 log k) subcases.

Two remarks are in order. First, in this branching step we in particular guess whenever for some cell one point is the extremal point in more than one directions, as then the extremal points corresponding to these directions will be guessed to be equal both in ≤_x and in ≤_y. Second, if cell and cell′ are not between the same two consecutive vertical lines, then the relation of the extremal points in cell and cell′ in the order ≤_x can be inferred and does not need to be guessed; similarly if cell and cell′ are not between the same two consecutive horizontal lines, their relation in the ≤_y order can be inferred.

In what follows we will use the information guessed in this step in the following specific scenario (see Figure 3): for every apx-supercell apxcell and direction Δ ∈ {top, bottom, left, right}, we will be interested in the relative order in ≤_x (if Δ ∈ {left, right}) or ≤_y (if Δ ∈ {top, bottom}) of the Δ-most extremal points of the cells in apxcell that share the Δ border with apxcell.

All the above branching steps lead to 2^Ω(k^2 log k) log n subcases in total. With each subcase, we proceed...
Let $\ell$ this task as a CSP instance with binary constraints and variable set.

Branching Step C that assigns to each cell the type in Corner filtering and corner constraints.

Recall that monotonicity constraints.

For every two consecutive Monotonicity constraints.

We start with two simple categories of constraints.

4.4 Simple filtering steps and constraints

to the next steps of the algorithm.

4.3 CSP formulation

Recall that $\zeta^{x,\text{apx}}$ and $\zeta^{y,\text{apx}}$ map the abstract vertical line set $X_{\text{lin}}^{\text{apx}}$ and horizontal line set $Y_{\text{lin}}^{\text{apx}}$, respectively, to the concrete integer coordinates and that we want to extend these functions to increasing functions $\zeta^x: X_{\text{lin}} \to \mathbb{N}$ and $\zeta^y: Y_{\text{lin}} \to \mathbb{N}$ such that $\{\zeta^x(\ell) \mid \ell \in X_{\text{lin}}^{\text{opt}}\}$ and $\{\zeta^y(\ell) \mid \ell \in Y_{\text{lin}}^{\text{opt}}\}$ is a separation. We phrase this task as a CSP instance with binary constraints and variable set $X_{\text{lin}}^{\text{opt}} \cup Y_{\text{lin}}^{\text{opt}}$, where we shall assign to each variable $\ell \in X_{\text{lin}}$ value of $\zeta^x(\ell)$ and analogous for $Y_{\text{lin}}$. The domains are initially defined as follows.

Let $\ell \in X_{\text{lin}}^{\text{opt}}$. Let $\ell_1$ be the maximum element of $X_{\text{lin}}^{\text{apx}}$ with $\ell_1 < \ell$ and let $\ell_2$ be the minimum element of $X_{\text{lin}}^{\text{apx}}$ with $\ell < \ell_2$. (Recall that here $<$ is the order of lines determined and defined after Branching Step B.) We define the domain $D_\ell$ of $\ell$ to be

$$D_\ell := \{a \in \mathbb{N} \mid \zeta_{\text{apx}}(\ell_1) < a < \zeta_{\text{apx}}(\ell_2) \land a \equiv 2 \pmod{3}\}.$$

We define the domain $D_\ell$ for each $\ell \in Y_{\text{opt}}^{\text{apx}}$ analogously. Note that, by the discretization properties, such domains can be computed in polynomial time.

To define the final CSP instance, we will in the following do two operations: introduce constraints and do filtering steps. We will introduce constraints in five different categories: monotonicity, corner, alternations, correct order of extremal points, and alternating lines. The filtering steps remove values from variable’s domains that represent situations that we know or have guessed to be impossible. To show correctness of the so-constructed reduction to CSP, observe that it suffices to define the constraints and conduct the filtering steps to ensure the following two properties:

**Soundness** — if in the current branch we have guessed all the information about $(X, Y)$ correctly, then the pair $(\zeta_{\text{opt}}^x, \zeta_{\text{opt}}^y)$ is a satisfying assignment to the constructed CSP instance (that is, the values of $(\zeta_{\text{opt}}^x, \zeta_{\text{opt}}^y)$ are never removed from the corresponding domains in the filtering steps and $(\zeta_{\text{opt}}^x, \zeta_{\text{opt}}^y)$ satisfies all introduced constraints).

**Completeness** — for a satisfying assignment $(\zeta_{\text{opt}}^x, \zeta_{\text{opt}}^y)$ to the final CSP instance, the pair $\{\zeta_{\text{opt}}^x(\ell) \mid \ell \in X_{\text{lin}}^{\text{opt}}\}, \{\zeta_{\text{opt}}^y(\ell) \mid \ell \in Y_{\text{lin}}^{\text{opt}}\}$ is a separation.

We now proceed to define the five categories of constraints and a number of filtering steps. For every introduced constraint and conducted filtering step, the soundness property will be straightforward. A tedious but relatively natural check will ensure that all introduced constraints of four categories together with the filtering steps ensure the completeness property. While introducing constraints, we will be careful to limit their number to a polynomial in $k$ and to ensure that every introduced constraint has a segment representation of constant or $\mathcal{O}(k)$ depth. This, together with the results of Section 3, prove Theorem 1.1.

4.4 Simple filtering steps and constraints

We start with two simple categories of constraints.

**Monotonicity constraints.** For every two consecutive $\ell_1, \ell_2 \in X_{\text{lin}}^{\text{opt}}$ or two consecutive $\ell_1, \ell_2 \in Y_{\text{lin}}^{\text{opt}}$, we add a constraint that the value of $\ell_1$ is smaller than the value of $\ell_2$.

It is clear that the above constraints maintain soundness. By Observation 2.6 every such constraint is of depth 1 and its segment representation can be computed in polynomial time. Furthermore, there are $\mathcal{O}(k)$ monotonicity constraints.

**Corner filtering and corner constraints.** Recall that $\delta: \text{Cells} \to \{0, 1, 2\}$ is the function guessed in Branching Step C that assigns to each cell the type in $\{0, 1, 2\}$ according to whether it contains points of $W_1$ (type 1), points of $W_2$ (type 1), or no points at all (type 0). We inspect every tuple of two vertical lines $p_1, p_2 \in X_{\text{lin}}$ with $p_1 < p_2$ and two horizontal lines $\ell_1, \ell_2 \in Y_{\text{lin}}$ with $\ell_1 < \ell_2$ such that

- there is no line of $X_{\text{lin}}^{\text{apx}}$ between $p_1$ and $p_2$ and there is no line of $Y_{\text{lin}}^{\text{apx}}$ between $\ell_1$ and $\ell_2$;
A tuple \((p_1, p_2, \ell_1, \ell_2)\) satisfying the conditions above is called an empty corner.

We would like to ensure that in the space area\((p_1, p_2, \ell_1, \ell_2)\) between \(p_1\) and \(p_2\) and between \(\ell_1\) and \(\ell_2\) (henceforth called the area of interest of the tuple \((p_1, p_2, \ell_1, \ell_2)\)) there are no points of \(W_1 \cup W_2\). Since at most two lines of \(\{p_1, p_2, \ell_1, \ell_2\}\) are from \(X^\text{apx} \cup Y^\text{apx}\), we can do it with either restricting domains of some variables or with a relatively simple binary constraint as described below. Herein, we distinguish the three cases of how many lines from \(X^\text{apx} \cup Y^\text{apx}\) there are in the tuple:

**Corner filtering.** Observe that the area of interest of the tuple \((p_1, p_2, \ell_1, \ell_2)\) is always contained in a single apx-supercell. If all lines of \(\{p_1, p_2, \ell_1, \ell_2\}\) are from \(X^\text{apx} \cup Y^\text{apx}\), then the area of interest of \((p_1, p_2, \ell_1, \ell_2)\) is the apx-supercell \(\text{apxcell}(p_1, \ell_1)\). If this apx-supercell contains at least one point of \(W_1 \cup W_2\), we reject the current branch.

If exactly one line of \(\{p_1, p_2, \ell_1, \ell_2\}\) is not from \(X^\text{apx} \cup Y^\text{apx}\), say \(\ell\), then we inspect all the values of \(D_\ell\) and delete those values for which there is some point of \(W_1 \cup W_2\) in the area of interest of \((p_1, p_2, \ell_1, \ell_2)\).

It is straightforward to see that, if all guesses were correct, then the above filtering steps do not remove any value of \((\zeta_X^\text{opt}, \zeta_Y^\text{opt})\) from the corresponding domains, that is, they preserve soundness.

**Corner constraints.** If exactly two lines of \(\{p_1, p_2, \ell_1, \ell_2\}\) are not from \(X^\text{apx} \cup Y^\text{apx}\), say \(\ell\) and \(\ell'\), then we add a constraint binding \(\ell\) and \(\ell'\) that allows only values \(x \in D_\ell\) and \(x' \in D_{\ell'}\) that leave the area of interest of \((p_1, p_2, \ell_1, \ell_2)\) empty.

It is straightforward to verify that, if all guesses were correct, the pair \((\zeta_X^\text{opt}, \zeta_Y^\text{opt})\) satisfies all introduced corner constraints, that is, soundness is preserved. We now consider the number of constraints and the running time of adding them. Indeed, as we will see below, some of the constraints above are superfluous and we can omit them.

**Lemma 4.1.** A corner constraint added for a tuple \((p_1, p_2, \ell_1, \ell_2)\) with exactly two lines from \(X^\text{apx} \cup Y^\text{apx}\) is of the form treated in Observation 2.3 and, consequently, is a conjunction of at most four constraints, each of depth at most 2, and the segment representations of these constraints can be computed in polynomial time.

**Proof.** Let \(\ell\) and \(\ell'\) be the two lines of \(\{p_1, p_2, \ell_1, \ell_2\}\) that are not from \(X^\text{apx} \cup Y^\text{apx}\) and let \(\text{apxcell}\) be the (abstract) apx-supercell containing the area of interest of \((p_1, p_2, \ell_1, \ell_2)\). The constraint asserting that no point of \((W_1 \cup W_2) \cap \text{apxcell}\) is in the area of interest of \((p_1, p_2, \ell_1, \ell_2)\) can be expressed as a conjunction over all \((x, y) \in (W_1 \cup W_2) \cap \text{apxcell}\) of the constraints \(C_{x,y}\) stating that \((x, y)\) is not in the area of interest. Constraint \(C_{x,y}\), in turn, can be expressed as \((x < \zeta_X(\ell)) \lor (y < \zeta_Y(\ell'))\) if \(\ell = \ell_1\) and \(\ell' = \ell_1\) and similarly if \(\ell = \ell_2\) and \(\ell' = \ell_2\). By Observation 2.3, a conjunction of such constraints \(C_{x,y}\) is a conjunction of at most four constraints, each of depth at most 2 and it follows from the simple form of these constraints that their segment representations can be computed in polynomial time.

Let us now bound the number of corner constraints that we need to add.

There are \(O(k^2)\) tuples \((p_1, p_2, \ell_1, \ell_2)\) for which \(p_1 \in X^\text{opt}\) and \(\ell_1 \in Y^\text{opt}\), as the choice of \(p_1\) and \(\ell_1\) already determines \(p_2\) and \(\ell_2\). Hence, by symmetry, there are \(O(k^2)\) tuples \((p_1, p_2, \ell_1, \ell_2)\) that contain one line of \(X^\text{opt}\) and one line of \(Y^\text{opt}\).

Consider now a tuple \((p_1, p_2, \ell_1, \ell_2)\) where \(p_1, p_2 \in X^\text{opt}\) and \(\ell_1, \ell_2 \in Y^\text{apx}\). Then \(\ell_1\) and \(\ell_2\) are two consecutive elements of \(Y^\text{apx}\); there are \(O(k)\) choices for them. If there is also an empty corner \((p_1, p_2, \ell_1, \ell_2)\) with \(p_2 < p_2'\), then the corner constraint for \((p_1, p_2, \ell_1, \ell_2)\), together with monotonicity constraints, implies the corner constraint for \((p_1, p_2, \ell_1, \ell_2)\). Hence, we can add only corner constraints for empty corners \((p_1, p_2, \ell_1, \ell_2)\) with maximal \(p_2\). In this manner, we add only \(O(k^2)\) corner constraints for empty corners \((p_1, p_2, \ell_1, \ell_2)\) with \(p_1, p_2 \in X^\text{opt}\). Similarly, we add only \(O(k^2)\) corner constraints for tuples \((p_1, p_2, \ell_1, \ell_2)\) with \(\ell_1, \ell_2 \in Y^\text{opt}\).

To sum up, we add \(O(k^2)\) corner constraints, each of depth at most 2.
4.5 Alternation of a situation

Outline. Unfortunately, monotonicity and corner constraints are not sufficient to ensure completeness. To see this, consider an apx-supercell apxcell($p_1, \ell_1$) with $p_2$ and $\ell_2$ being the successors of $p_1$ and $\ell_1$ in $X_{lin}^{apx}$ and $Y_{lin}^{apx}$, respectively. If there is exactly one line $p \in X_{lin}^{opt}$ between $p_1$ and $p_2$ and exactly one line $\ell \in Y_{lin}^{opt}$ between $\ell_1$ and $\ell_2$, then any of the cells cell($p_1, \ell_1$), cell($p, \ell_1$), cell($p_1, \ell$), or cell($p, \ell$) that is guessed to be empty by $\delta$ is taken care of by the corner constraint for the empty corner ($p_1, p, \ell_1, \ell$), ($p_2, \ell$, $\ell_1, \ell_2$), and ($p, p_2, \ell, \ell_2$), respectively. More generally, the corner constraints and other filtering performed above takes care of empty cells contained in area($p_1, p_2, \ell_1, \ell_2$) if there is at most one line of $X_{lin}^{opt}$ between $p_1$ and $p_2$ and at most one line of $Y_{lin}^{opt}$ between $\ell_1$ and $\ell_2$. However, consider a situation in which there are, say, three lines $\ell^1, \ell^2, \ell^3 \in Y_{lin}^{opt}$ between $\ell_1$ and $\ell_2$ and one line $p \in X_{lin}^{opt}$ between $p_1$ and $p_2$ (see Figure 4). If $\delta$(cell($p, \ell^2$)) = 0 but $\delta$(cell($p, \ell^1$)) $\neq$ 0 and $\delta$(cell($p, \ell^3$)) $\neq$ 0, then the cell cell($p, \ell^2$) is not contained in the area of interest of any of the empty corners and the corner constraints are not sufficient to ensure that cell($p, \ell^2$) is left empty.

The problem in formulating the constraints for such sandwiched cells is that the possible values for the enclosing optimal lines depend not only on the points inside the current apx-supercell, but also on the way points are to be separated possibly outside of the current apx-supercell. We begin to disentangle this intricate and non-local relationship by first focusing on lines that ensure correct separation of points within the current apx-supercell. We will call opt-lines ensuring such separation alternating, and their positions give rise to alternation constraints and alternating lines constraints.

We perform what follows in both dimensions, left/right and top/bottom. For the sake of clarity of description, we present description in the direction “left/right” (we found introducing an abstract notation of directions too cumbersome, given the complexity of the arguments). However, the same steps and arguments apply to the and to the “top/bottom” directions, when we swap the roles of x- and y-axes.

Definitions. Let $p_1, p_2$ be two consecutive elements of $X_{lin}^{opt}$ that are not consecutive elements of $X_{lin}$ (i.e., there is at least one line of $X_{lin}^{apx}$ between them). Let $\ell_1, \ell_2$ be two consecutive elements of $Y_{lin}^{apx}$. The tuple $\sigma = (p_1, p_2, \ell_1, \ell_2)$ is called a situation. See Fig. 5 for an example. Let $L_\sigma$ be the set of lines from $Y_{lin}^{opt}$ that are between $\ell_1$ and $\ell_2$. Let $L'_\sigma := L_\sigma \cup \{\ell_1\}$. For both $i = 1, 2$ let $p'_i$ be the maximum element of $X_{lin}^{apx}$ that
In the natural manner: Put \( \ell_1 = \ell_0 = \ell_1 \) \( \ell_2 \)
\( \ell_3 \)
\( \ell_4 \)
\( \ell_5 \)
\( \ell_6 \)

is smaller than \( p_i \).

For each \( \ell \in L'_\sigma \), we define \( \text{area}(\ell) := \text{area}(p_1, p_2, \ell, \ell') \), where \( \ell' \) is the successor of \( \ell \) in \( L_\sigma \cup \{\ell_1, \ell_2\} \)
Note that \( \text{area}(\ell) = \text{optcell}(p_1, \ell) \) for every \( \ell \in L'_\sigma \) except for possibly \( \ell = \ell_1 \) and \( \ell \) being the maximum element of \( L_\sigma \). However, if \( \ell = \ell_1 \) then \( \text{area}(\ell) \) is contained in \( \text{optcell}(p_1, \ell'_1) \) where \( \ell'_1 \) is the predecessor of the minimum element of \( L_\sigma \) in \( Y_{\text{opt}} \cup \{ (\zeta, ap)^{-1}(1) \} \), and if \( \ell \) is the maximum element of \( L_\sigma \), then \( \text{area}(\ell) \) is contained in \( \text{optcell}(p_1, \ell) \).

Recall that \( \delta: \text{Cells} \rightarrow \{0, 1, 2\} \) is the function guessed in Branching Step C that assigns to each cell its content type. By the above inclusion-property of cells, we can extend the function \( \delta \) to \( \{\text{area}(\ell) \mid \ell \in L'_\sigma \} \) in the natural manner: Put \( \delta(\text{area}(\ell)) = 0 \) if every cell \( \text{cell} \) contained in \( \text{area}(\ell) \) satisfies \( \delta(\text{cell}) = 0 \) and, otherwise, \( \delta(\text{area}(\ell)) \) is defined as the unique nonzero value attained by \( \delta(\text{cell}) \) for \( \text{cell} \) contained in \( \text{area}(\ell) \).

Note that the values of \( \delta(\text{cell}) \) for \( \text{cell} \) contained in \( \text{area}(\ell) \) cannot attain both values 1 and 2, as they are all contained in one and the same \( \text{opt-supercell} \).

An element \( \ell \in L'_\sigma \) is alternating if \( \delta(\text{area}(\ell)) \neq 0 \), the maximum element \( \ell' \in L'_\sigma \) with \( \ell' < \ell \) and \( \delta(\text{area}(\ell')) \neq 0 \) exists, and \( \delta(\text{area}(\ell)) \neq \delta(\text{area}(\ell')) \). Let \( \bar{L}_\sigma \) be the set of alternating elements of \( L'_\sigma \) and let \( \bar{L}'_\sigma = \bar{L}_\sigma \cup \{\ell_0\} \) where \( \ell_0 \) is the minimum element of \( L'_\sigma \) with \( \delta(\text{area}(\ell_0)) \neq 0 \). We define \( \bar{L}_\sigma = \bar{L}'_\sigma = \emptyset \) if each element \( \ell \in L'_\sigma \) has \( \delta(\text{area}(\ell)) = 0 \), that is, every cell \( \text{cell} \) contained in \( \text{area}(p_1, p_2, \ell_1, \ell_2) \) satisfies \( \delta(\text{cell}) = 0 \).

Consider the sequence \( S_\sigma \) consisting of values \( \delta(\text{area}(\ell)) \) for \( \ell \in L'_\sigma \), ordered in the increasing order of the corresponding lines in \( L'_\sigma \). Similarly, \( \bar{S}_\sigma \) is a sequence consisting of values \( \delta(\text{area}(\ell)) \) for \( \ell \in \bar{L}'_\sigma \), ordered in the increasing order of the corresponding lines in \( \bar{L}'_\sigma \).

The alternation of a situation \( \sigma \) is the length of the sequence \( \bar{S}_\sigma \). Observe that equivalently we can define \( \bar{S}_\sigma \) as the maximum length of a subsequence of alternating 1s and 2s in \( S_\sigma \) (the sequence may start either with a 2 or with a 1).

**Observations.** Intuitively, in what follows we focus on alternating lines as they are the ones that separate \( W_1 \) from \( W_2 \) within the area bounded by \( p_1, p_2, \ell_1, \) and \( \ell_2 \). The introduced constraints are not meant to exactly focus that the content of every cell is as guessed by the function \( \delta \), but only that the alternating lines are placed correctly. See Figure 5.

We now make use of the branching steps to limit possible alternations.

---

Figure 5: A situation of alternation 6. The lines of \( \bar{L}'_\sigma \) are denoted with \( \ell^i \), \( 1 \leq i \leq 6 \).
Lemma 4.2. Assume that all guesses in the branching steps were correct regarding the solution $(X,Y)$. For each situation $\sigma = (p_1,p_2,\ell_1,\ell_2)$, the alternation equals 0, 1, or it is an even positive integer. If the alternation is at least four, then $\delta(\text{apxcell}(p'_1,\ell_1))$ and $\delta(\text{apxcell}(p'_2,\ell_1))$ are different and both are nonzero.

Proof. Let $\ell$ and $\ell'$ be the minimum and maximum elements of $L_{\sigma}$, respectively.

Suppose that the contrary of the first statement holds. Due to symmetry between $W_1$ and $W_2$, we may assume without loss of generality that $S_{\sigma} = (12)^r 1$ for some $r \geq 1$. This in particular implies $|L_{\sigma}| = 2r \geq 2$, so $|L_{\sigma}| \geq 2$. Let $p \in X_{\text{lin}}^{\text{apx}}$ with $p'_1 \leq p \leq p'_2$, $\delta(\text{apxcell}(p,\ell_1)) = 2$, and such that some point of $W_2$ in $\text{apxcell}(p,\ell_1)$ lies between $\zeta^{\text{opt}}_X(p_1)$ and $\zeta^{\text{opt}}_X(p_2)$; $p$ exists as $r \geq 1$. Let $(x,y) \in \text{apxcell}(p,\ell_1) \cap W_2$ be defined as follows: if $p = p'_1$, then $(x,y)$ is the rightmost element of $\text{apxcell}(p,\ell_1) \cap W_2$, and if $p > p'_1$, then $(x,y)$ is the leftmost element of $\text{apxcell}(p,\ell_1) \cap W_2$. The point $(x,y)$ is an extremal point in the apx-supercell $\text{apxcell}(p,\ell_1)$. Observe that, by the choice of $p$, coordinate $x$ lies between $\zeta^{\text{opt}}_X(p_1)$ and $\zeta^{\text{opt}}_X(p_2)$ while, by the structure of $S_{\sigma}$, coordinate $y + 1$ lies between $\zeta^{\text{opt}}_Y(\ell)$ and $\zeta^{\text{opt}}_Y(\ell')$. This again contradicts the correctness of the guess at Branching Step A. Thus the first statement holds.

For the second statement, the reasoning is similar. For the sake of contradiction, suppose that the contrary of the second statement holds. Then, due to symmetry between $W_1$ and $W_2$, we may assume without loss of generality that

$$
\delta(\text{apxcell}(p'_1,\ell_1)), \delta(\text{apxcell}(p'_2,\ell_1)) \in \{0,1\}.
$$

(4)

Since the alternation is at least four, we have $|L_{\sigma}| \geq |W_2| \geq 3$. Let $W$ be the set of elements of $W_2$ between $p_1$ and $p_2$ and between $\ell_1$ and $\ell_2$. By Eq. (4) and since $S_{\sigma}$ contains at least one 2, we have $W \neq \emptyset$. If $S_{\sigma} = (12)^r$ for some $r \geq 2$, then let $(x,y)$ be the bottommost element of $W$ and otherwise, if $S_{\sigma} = (21)^r$, then let $(x,y)$ be the topmost element of $W$. Observe that $(x,y)$ lies in $\text{apxcell}(p,\ell_1)$ for some $p \in X_{\text{lin}}^{\text{apx}}$ with $p'_1 < p < p'_2$ and is the bottommost or topmost, respectively, element of $\text{apxcell}(p,\ell_1)$. Furthermore, $y + 1$ lies between $\zeta^{\text{opt}}_Y(\ell)$ and $\zeta^{\text{opt}}_Y(\ell')$. This again contradicts the correctness of the guess at Branching Step A.\]

An astute reader can observe (and it will be proven formally later) than in a situation of alternation at most two, the corner constraints and filtering steps are sufficient to ensure completeness. Thus, we introduce below alternation and alternating-lines constraints only for situations with alternation at least four. Henceforth we assume that the studied situation $\sigma = (p_1,p_2,\ell_1,\ell_2)$ has alternation at least four. By symmetry and Lemma 4.2, we can assume that $\delta(\text{apxcell}(p'_1,\ell_1)) = 1$, $\delta(\text{apxcell}(p'_2,\ell_1)) = 2$ (otherwise we swap the roles of $W_1$ and $W_2$) and additionally that $S_{\sigma} = (12)^r$ for some $r \geq 2$ (otherwise we reflect the instance on an arbitrary horizontal line). We remark also that the reflection step above may require adding +1 to the depth of the introduced constraints; this will not influence the asymptotic number and total depth of introduced constraints.

Assume now we are given a situation $\sigma = (p_1,p_2,\ell_1,\ell_2)$ and fixed values $\zeta^{\text{opt}}_X(p_1) = x_1$ and $\zeta^{\text{opt}}_X(p_2) = x_2$. With these values, let $\text{pts}_x(x_1,x_2)$ be the set of points of $W_1 \cup W_2$ in the area bounded by $p_1, p_2, \ell_1, \ell_2$ (recall that $\ell_1, \ell_2 \in Y_{\text{lin}}^{\text{apx}}$). Define the sequence $S(x_1,x_2) \in \{1,2\}^*$ as follows. Let $(w_1,\ldots,w_s)$ be the sequence of all points from $\text{pts}_x(x_1,x_2)$ such that for each $i \in [s]$ we have $w_i \leq y_{w_i+1}$. Then, $S(x_1,x_2) := (\alpha(w_1),\ldots,\alpha(w_s))$ where $\alpha(w_i) = \beta \in \{1,2\}$ if $w_i \in W_\beta$. A block is a set of points in $\text{pts}_x(x_1,x_2)$ that correspond to a maximal block of consecutive equal values in $S(x_1,x_2)$. The definition of blocks is depicted in Figure 6.

The sequence $S(x_1,x_2)$ is the subsequence of the sequence $S(x_1,x_2)$ that consists of the first element and all elements whose predecessor is a different element (i.e., $S(x_1,x_2)$ contains one element for every maximal block of equal elements in $S(x_1,x_2)$).

We define the alternation of points $\text{pts}_x(x_1,x_2)$ as follows. If there are two points in $\text{pts}_x(x_1,x_2)$ with the same $y$-coordinate but one from $W_1$ and one from $W_2$, the alternation is $+\infty$. Otherwise, the alternation of $\text{pts}_x(x_1,x_2)$ is the number of maximal blocks of consecutive equal values in $S(x_1,x_2)$, that is, the length of $S(x_1,x_2)$.

We have the following straightforward observation (recall that $p_1$ and $p_2$ are consecutive elements of $X_{\text{lin}}^{\text{opt}}$).
Observation 4.3. Let $\sigma = (p_1, p_2, \ell_1, \ell_2)$ be a situation. If the guesses in all branching steps were correct regarding the solution $(X, Y)$ and $x_i = \zeta_{X}^{\text{opt}}(p_i)$ for $i = 1, 2$, then the alternations of $\text{pts}_\sigma(x_1, x_2)$ and of $\sigma$ are equal and $S(x_1, x_2) = \bar{S}_\sigma$. In particular, the alternation of $\text{pts}_\sigma(x_1, x_2)$ is finite.

We say that the pair $(x_1, x_2) \in D_{p_1} \times D_{p_2}$ fits the alternation of the situation $\sigma = (p_1, p_2, \ell_1, \ell_2)$ if the conclusion of Observation 4.3 is satisfied for $\text{pts}_\sigma(x_1, x_2)$ and the situation $\sigma$, that is, $S(x_1, x_2) = \bar{S}_\sigma$ and the alternation of $\text{pts}_\sigma(x_1, x_2)$ is finite. Observe the following.

Observation 4.4. Let $(x_1, x_2) \in D_{p_1} \times D_{p_2}$ be a pair that does not fit the situation $\sigma = (p_1, p_2, \ell_1, \ell_2)$. Then, one of the following is true:

(a) The alternation of $\text{pts}_\sigma(x_1, x_2)$ is finite and smaller than the alternation of $\sigma$. That is, $\bar{S}(x_1, x_2)$ is a proper subsequence of the sequence $\bar{S}_\sigma$.

(b) The alternation of $\text{pts}_\sigma(x_1, x_2)$ is infinite or not smaller than the alternation of $\sigma$. That is, $\bar{S}(x_1, x_2)$ is not a subsequence of the sequence $\bar{S}_\sigma$.

For a pair $(x_1, x_2) \in D_{p_1} \times D_{p_2}$ that does not fit the situation $\sigma = (p_1, p_2, \ell_1, \ell_2)$, we say that $(x_1, x_2)$ is of Type [a] or Type [b] depending on which case of Observation 4.4 it falls into.

Observe that for every $(x_1, x_2), (x_1', x_2') \in D_{p_1} \times D_{p_2}$ with $x_1 \leq x_1'$ and $x_2 \leq x_2'$ the set $\text{pts}_\sigma(x_1', x_2')$ is a subset of the set $\text{pts}_\sigma(x_1, x_2)$, so $S(x_1', x_2')$ is a subsequence of $S(x_1, x_2)$ and thus $\bar{S}(x_1', x_2')$ is a subsequence of $\bar{S}(x_1, x_2)$. Hence, we have the following.

Observation 4.5. Let $(x_1, x_2) \in D_{p_1} \times D_{p_2}$ be a pair that does not fit the situation $\sigma = (p_1, p_2, \ell_1, \ell_2)$. If $(x_1, x_2)$ is of Type [a] and $(x_1', x_2') \in D_{p_1} \times D_{p_2}$ is such that $x_1 \leq x_1'$ and $x_2 \leq x_2'$, then $(x_1', x_2')$ is of Type [a] too (in particular, does not fit $\sigma$). Similarly, if $(x_1, x_2)$ is of Type [b] and $(x_1', x_2') \in D_{p_1} \times D_{p_2}$ is such that $x_1' \leq x_1$ and $x_2 \leq x_2'$, then $(x_1', x_2')$ is of Type [b] too (and, again, does not fit $\sigma$).
4.5.1 Filtering for correct alternation

We exhaustively perform the following filtering operation for each situation \( \sigma = (p_1, p_2, \ell_1, \ell_2) \): If there exists \( x_1 \in D_{p_1} \) such that there is no \( x_2 \in D_{p_2} \) such that \((x_1, x_2)\) fits \( \sigma \), we remove \( x_1 \) from \( D_{p_1} \) and symmetrically, if there exists \( x_2 \in D_{p_2} \) such that there is no \( x_1 \in D_{p_1} \) such that \((x_1, x_2)\) fits \( \sigma \), we remove \( x_2 \) from \( D_{p_2} \).

Henceforth we assume that for every \( x_1 \in D_{p_1} \) there is at least one \( x_2 \in D_{p_2} \) such that \((x_1, x_2)\) fits \( \sigma \) and for every \( x_2 \in D_{p_2} \) there is at least one \( x_1 \in D_{p_1} \) such that \((x_1, x_2)\) fits \( \sigma \). Clearly, if all the branching steps made a correct guesses, we do not remove neither \( \zeta_{x, \text{opt}}(p_1) \) from \( D_{p_1} \) nor \( \zeta_{x, \text{opt}}(p_2) \) from \( D_{p_2} \). That is, this filtering step is sound. It is not hard to see that it can be carried out in polynomial time.

For the alternating lines constraints in Section 4.7 we need the following observation on the structure of the remaining values. Consider a value \( x_1 \in D_{p_1} \) for the line variable \( p_1 \in X_{\text{lin}} \). Observation 4.5 implies that set of \( x_2 \in D_{p_2} \) such that \((x_1, x_2)\) fit \( \sigma \) forms a segment in \( D_{p_2} \). Let \( x_2^\sup(x_1) \) and \( x_2^\inf(x_1) \) be the minimum and maximum values \( x_2 \in D_{p_2} \) for which \((x_1, x_2)\) fits \( \sigma \). Similarly, for a value \( x_2 \in D_{p_2} \), let \( x_1^\sup(x_2) \) and \( x_1^\inf(x_2) \) be the minimum and maximum values \( x_1 \in D_{p_1} \) for which \((x_1, x_2)\) fits \( \sigma \). Observe that \( x_1^\sup \) defines a function \( D_{p_1} \to D_{p_2} \) and analogously for \( x_1^\inf \), \( x_2^\sup \), and \( x_2^\inf \). Note that Observation 4.5 implies the following.

**Observation 4.6.** The functions \( x_1^\sup \), \( x_1^\inf \), \( x_2^\sup \), and \( x_2^\inf \) are nondecreasing.

4.5.2 Alternation constraints

Observation 4.3 asserts that the values \((x_1, x_2)\) of variables \( \ell_1 \) and \( \ell_2 \) in the solution \((X, Y)\) fit the situation \( \sigma \). This motivates adding the following constraints. For every situation \( \sigma = (p_1, p_2, \ell_1, \ell_2) \) of alternation at least four we add a constraint binding \( p_1 \) and \( p_2 \) that allows only pairs of values \((x_1, x_2)\) that fit the situation \( \sigma \).

Observation 4.3 asserts that the assignment \( \zeta_{x, \text{opt}} \cup \zeta_{y, \text{opt}} \) satisfies all alternation constraints. Clearly, there are \( O(k^2) \) alternation constraints, as the choice of \( p_1 \) and \( \ell_1 \) defines the situation \( \sigma = (p_1, p_2, \ell_1, \ell_2) \).

We now prove that a single alternation constraint is a conjunction of two constraints of bounded depth.

**Lemma 4.7.** For a situation \( \sigma = (p_1, p_2, \ell_1, \ell_2) \) of alternation at least four, the alternation constraint binding \( p_1 \) and \( p_2 \) is equivalent to a conjunction of two constraints, each with a segment representation of depth 1. Moreover, the latter conjunction of constraints and their segment representations can be computed in polynomial time.

**Proof.** By Observation 4.4 the discussed alternation constraint is a conjunction of a constraint \( ^a(\zeta_{x, \text{opt}}(p_1), \zeta_{x, \text{opt}}(p_2)) \) is not of Type \( [a] \) and a constraint \( ^a(\zeta_{x, \text{opt}}(p_1), \zeta_{x, \text{opt}}(p_2)) \) is not of Type \( [b] \). By Observation 4.5 the constraint \( ^a(\zeta_{x, \text{opt}}(p_1), \zeta_{x, \text{opt}}(p_2)) \) is not of Type \( [a] \) is a conjunction, over all pairs \((x_1, x_2)\) \( \in D_{p_1} \times D_{p_2} \) of Type \( [a] \) of a constraint \( \zeta_{x, \text{opt}}(p_1) < x_1 \) \( \vee \) \( \zeta_{x, \text{opt}}(p_2) > x_2 \). Such a conjunction can be represented with a segment representation of depth 1 due to Observation 2.5. Similarly, by Observation 4.5 again the constraint \( ^a(\zeta_{x, \text{opt}}(p_1), \zeta_{x, \text{opt}}(p_2)) \) is not of Type \( [b] \) is a conjunction, over all pairs \((x_1, x_2)\) \( \in D_{p_1} \times D_{p_2} \) of Type \( [b] \) of a constraint \( \zeta_{x, \text{opt}}(p_1) > x_1 \) \( \vee \) \( \zeta_{x, \text{opt}}(p_2) < x_2 \). Again, such a conjunction can be represented with a segment representation of depth 1 due to Observation 2.5. This finishes the proof of the lemma.

Consequently, by adding alternation constraints we add \( O(k^2) \) constraints, each of depth 1.

4.6 Filtering for correct orders of extremal points

Unfortunately, alternation constraints are still not enough to ensure completeness—we need to restrict the places where the alternation occurs further in order to be able to formulate a CSP of the form described in Section 3. Consider a situation \( \sigma = (p_1, p_2, \ell_1, \ell_2) \), fixing a position for \( p_1 \), and moving the position for \( p_2 \) in increasing \( \leq_x \)-order over the positions where the correct alternation is obtained. This gives sequences of possible positions for the horizontal lines that ensure the correct alternation. However, intuitively, it is possible for these positions to jump within the \( \leq_x \)-order in a noncontinuous fashion, from top to bottom and back. We have no direct way of dealing with such discontinuity in the CSP of the form in Section 3. To avoid this behaviour, we now make crucial use of the information we have guessed in Branching Step D.
and E. In combination with Observations 4.3, 4.4, and 4.5, we can smooth the admissible positions for the horizontal lines that give the correct alternation.

Recall that we have assumed without loss of generality that \( \delta(apxcell(p_1', \ell_1)) = 1 \) and \( \delta(apxcell(p_2', \ell_1)) = 2 \). Thus, having fixed the value \( x_1 \in D_{p_1} \) of \( \zeta^{x,\text{opt}}(p_1) \), the set of points from \( W_1 \) in \( pts_\sigma(x_1, x_2) \) is fixed, regardless of the value \( x_2 \in D_{p_2} \) of \( \zeta^{x,\text{opt}}(p_2) \). Furthermore:

**Observation 4.8.** Let \( x_1 \in D_{p_1} \). For every \( x_2 \in D_{p_2} \), with \( x_2^\leftarrow (x_1) \leq x_2 \leq x_2^\rightarrow (x_1) \), we have

\[
pts_\sigma(x_1, x_2^\leftarrow (x_1)) \cap W_2 \subseteq pts_\sigma(x_1, x_2) \cap W_2 \subseteq pts_\sigma(x_1, x_2^\rightarrow (x_1)) \cap W_2.
\]

Even further, the following important observation states that the partition of \( W_1 \cap pts_\sigma(x_1, x_2) \) into blocks does not depend on \( x_2 \).

**Observation 4.9.** Let \( x_1 \in D_{p_1} \). Then, the partition of the points of \( W_1 \cap pts_\sigma(x_1, x_2) \) into blocks is the same for any choice of \( x_2 \in D_{p_2} \), with \( x_2^\leftarrow (x_1) \leq x_2 \leq x_2^\rightarrow (x_1) \). Symmetrically, let \( x_2 \in D_{p_2} \). Then, the partition of the points of \( W_2 \cap pts_\sigma(x_1, x_2) \) into blocks is the same for any choice of \( x_1 \in D_{p_1} \), with \( x_1^\leftarrow (x_2) \leq x_1 \leq x_1^\rightarrow (x_2) \).

**Proof.** We prove only the first statement, the second one is symmetrical. Fix two integers \( x_2^\leftarrow (x_1) \leq x_2 \leq x_2^\rightarrow (x_1) \). Then the sequence \( S(x_1, x_2) \) is a subsequence of \( S(x_1, x_2') \) that contains the same number of 1s; they differ only in the number of 2s. Since both \((x_1, x_2')\) and \((x_1, x_2)\) fit \( \sigma \), \( S(x_1, x_2) = \tilde{S}(x_1, x_2') \). Hence, the maximal sequences of consecutive 1s in \( S(x_1, x_2) \) and \( S(x_1, x_2') \) are the same. Since set of points from \( W_1 \) in \( pts_\sigma(x_1, x_2) \) and \( pts_\sigma(x_1, x_2') \) are the same, the statement follows.

Observation 4.9 allows us to make the following filtering step for situation \( \sigma \) using the information guessed in Branching Steps D and E. Informally, in these branching steps we have guessed for each point for which cell it can be an extremal point. Since the blocks of \( W_1 \cap pts_\sigma(x_1, x_2) \) are fixed once \( x_1 \) is fixed, some of the extremal points are fixed, and we can now remove values from \( D_{p_1} \) for which this guess would be incorrect. Similar for \( x_2 \). The formal filtering step works as follows.

Recall that \( \tilde{S}(x_1, x_2^\leftarrow (x_1)) = (12)^r \) for some \( r \in \mathbb{N} \), \( r \geq 2 \). Let \( \ell^1, \ell^2, \ldots, \ell^r \) be the elements of \( \tilde{I}_\sigma \) in increasing order (i.e., \( \ell^1 \leq \ell^2 \leq \cdots \leq \ell^r \)), cf. Figure 3.

For each \( x_1 \in D_{p_1} \), let \( B^1_\sigma(x_1), B^2_\sigma(x_1), \ldots, B^r_\sigma(x_1) \) be the partition of \( pts_\sigma(x_1, x_2^\leftarrow (x_1)) \cap W_1 \) into blocks in the increasing order of \( \ell^1 \). Similarly, for each \( x_2 \in D_{p_2} \), let \( B^1_\sigma(x_2), B^2_\sigma(x_2), \ldots, B^r_\sigma(x_2) \) be the partition of \( pts_\sigma(x_2^\leftarrow (x_2), x_2) \cap W_2 \) into blocks in the increasing order of \( \ell^r \).

For each \( i \in [r] \), let \( \text{leader}^i_\sigma(x_1) \) be the rightmost element of \( B^i_\sigma(x_1) \) and let \( \text{leader}^i_\sigma(x_2) \) be the leftmost element of \( B^i_\sigma(x_2) \). Below we call these elements \textit{leaders}.

Assume that all branching steps made correct guesses regarding the solution \((X, Y)\) and consider \( x_1^{X, Y} := \zeta^{x,\text{opt}}(p_1) \), \( x_2^{X, Y} := \zeta^{x,\text{opt}}(p_2) \). Then, for every \( i \in [r] \), by the definition of alternating lines, the \( y \)-coordinate \( \zeta^{Y,\text{opt}}(\ell^i) \) lies between the \( y \)-coordinates of the points of \( B^i_\sigma(x_1^{X, Y}) \) and \( B^2_\sigma(x_2^{X, Y}) \) and, for every \( i \in [r] \), the \( y \)-coordinate \( \zeta^{Y,\text{opt}}(\ell^{i-1}) \) lies between the \( y \)-coordinates of the points of \( B^{i-1}_\sigma(x_2^{X, Y}) \) and \( B^i_\sigma(x_1^{X, Y}) \).

Also, for every \( i \in [r] \), the element \( \text{leader}^i_\sigma(x_1^{X, Y}) \) is the rightmost element of its cell and \( \text{leader}^i_\sigma(x_2^{X, Y}) \) is the leftmost element of its cell.

Fix \( i \in [r] \). We now observe that we can deduce from the information guessed in the branching steps to which cell the element \( \text{leader}^i_\sigma(x_1^{X, Y}) \) belongs. Indeed, \( B^i_\sigma(x_1^{X, Y}) \) consists of the cells \( \text{cell}(p, \ell) \) for all \( p \in X_{\infty} \) and \( \ell \in Y_{\infty} \) with \( p_1 \leq p < p_2 \) and \( \ell^{i-1} \leq \ell < \ell^i \). At Branching Step C we have guessed which of these cells are empty and which contain some element of \( W_1 \): We expect that \( \delta(\text{cell}(p, \ell)) = \{0, 1\} \) for every such pair \( (p, \ell) \) as above and \( \delta(\text{cell}(p, \ell)) = 1 \) for at least one such pair; we reject the current branch if this is not the case. The information guessed at Branching Step B allows us to infer

- the cell \( \text{cell}^i_1 \in \{\text{cell}(p, \ell) \mid p_1 \leq p < p_2 \land \ell^{i-1} \leq \ell < \ell^i\} \) that contains \( \text{leader}^i_\sigma(x_1^{X, Y}) \); and
- the relative order in \( \leq_\sigma \) of the elements \( \text{leader}^i_\sigma(x_1^{X, Y}) \) for \( 1 \leq i \leq r \).

Observe that, by Observation 4.9, given \( x_1 \in D_{p_1} \), we can in polynomial time compute whether the above two properties hold. We remove from \( D_{p_1} \) all values \( x_1 \) for which the order discussed in the second point above is
not as expected. Also, if the information guessed at Branching Step D is correct, we have \( \phi(leader_2^j(x_1^{X,Y})) = cell_1^j \). We remove from \( D_{p_1} \) all values \( x_1 \) for which there exists \( i \in [r] \) with \( \phi(leader_1^i(x_1)) \neq cell_1^i \). It is clear that this filtering step is sound and, as mentioned, it can be carried out in polynomial time.

We perform symmetrical analysis with the elements \( leader_2^i(x_2^{X,Y}) \). That is, the information guessed at Branching Step E allows us to infer

- the cell \( cell_2^i \in \{cell(p, \ell) \mid p_1 \leq p < p_2 \land \ell^{2i} \leq \ell < \ell^{2i+1}\} \) (with \( \ell^{2r+1} = \ell_2 \)) that contains \( leader_2^i(x_2^{X,Y}) \);
- and

- the relative order in \( \leq_x \) of the elements \( leader_2^i(x_2^{X,Y}) \) for \( i \in [r] \).

We remove from \( D_{p_2} \) all values \( x_2 \) for which the relative order in \( \leq_x \) of the elements \( leader_2^i(x_2) \) for \( i \in [r] \) is not as expected above. Also, if the information guessed at Branching Step D is correct, we have \( \phi(leader_2^i(x_2^{X,Y})) = cell_2^i \). We remove from \( D_{p_2} \) all values \( x_2 \) for which there exists \( i \in [r] \) with \( \phi(leader_2^i(x_2)) \neq cell_2^i \).

### 4.7 Alternating lines constraints

In the previous section we smoothed the possible positions for horizontal lines where the guessed alternation occurs. This enables us now to introduce constraints that describe these positions and have a form that is suitable for the type of CSP [3].

Fix a situation \( \sigma = (p_1, p_2, \ell_1, \ell_2) \) of alternation \( a \geq 4 \) and let \( r = a/2 \). Recall the definitions of the lines \( \ell_1, \ell_2, \ldots, \ell_r \), blocks \( B^1_1(\cdot), B^1_2(\cdot), \ldots, B^1_r(\cdot) \), and blocks \( B^2_1(\cdot), B^2_2(\cdot), \ldots, B^2_r(\cdot) \) from the previous section. We introduce the following constraints.

1. For every \( i \in [r] \), we introduce a constraint binding \( p_1 \) and \( \ell^{2i} \) that asserts that \( \zeta_{y, \text{opt}}(\ell^{2i}) \) is larger than the largest \( y \)-coordinate of an element of \( B^1_i(\zeta_{x, \text{opt}}(p_1)) \) (i.e., the line \( \ell^{2i} \) is above the block \( B^1_i \)).
2. For every \( i \in [r] \setminus \{1\} \), we introduce a constraint binding \( p_1 \) and \( \ell^{2i-1} \) that asserts that \( \zeta_{y, \text{opt}}(\ell^{2i-1}) \) is smaller than the smallest \( y \)-coordinate of an element of \( B^1_i(\zeta_{x, \text{opt}}(p_1)) \) (i.e., the line \( \ell^{2i-1} \) is below the block \( B^1_i \)).
3. For every \( i \in [r] \), we introduce a constraint binding \( p_2 \) and \( \ell^{2i} \) that asserts that \( \zeta_{y, \text{opt}}(\ell^{2i}) \) is smaller than the smallest \( y \)-coordinate of an element of \( B^2_i(\zeta_{x, \text{opt}}(p_2)) \) (i.e., the line \( \ell^{2i} \) is below the block \( B^2_i \)).
4. For every \( i \in [r-1] \), we introduce a constraint binding \( p_2 \) and \( \ell^{2i+1} \) that asserts that \( \zeta_{y, \text{opt}}(\ell^{2i+1}) \) is larger than the largest \( y \)-coordinate of an element of \( B^2_i(\zeta_{x, \text{opt}}(p_2)) \) (i.e., the line \( \ell^{2i+1} \) is above the block \( B^2_i \)).

We call the above constraints alternating-lines constraints. Again, the soundness property of the new constraints is straightforward. We now prove that the alternating lines constraints are of bounded depth (in the sense of Section [3]) and that a corresponding representation can be computed in polynomial time. This is the intuitive statement behind the following highly nontrivial lemma, whose proof spans the rest of this subsection.

**Lemma 4.10.** Let \( \sigma = (p_1, p_2, \ell_1, \ell_2) \) be a situation of alternation \( a \geq 4 \) and let \( r = a/2 \). Assume that \( \delta(apxcell(p_i', \ell_i)) = i \) for \( i = 1, 2 \), where \( p_i' \) is the predecessor of \( p_i \) in \( X_{apx}^\infty \), and that \( S\sigma = (12)^r \). Then one can in polynomial time compute two rooted trees \( T_j \) for \( j = 1, 2 \) with \( \text{leaves}(T_j) = \{v_j^1, v_j^2, \ldots, v_j^r, w_j^1, w_j^2, \ldots, w_j^r\} \) and \( |V(T_j)| = O(r) \), two families of segment reversions \( G_j = (g_{j,v})_{v \in V(T_j) \setminus \text{root}(T_j)} \) for \( j = 1, 2 \), and four families of downwards-closed relations \( R_j^{y, z, w, \mu} \) for \( j = 1, 2, 3, 4 \) such that the following holds. For every \( i \in [r] \) and \( j = 1, 2 \), let \( v_j^i = w_{j,1}, w_{j,2}, \ldots, w_{j,j} = \text{root}(T_j) \) be the nodes on the path from \( v_j^i \) to \( \text{root}(T_j) \) in the tree \( T_j \) and let \( w_j^i = z_{j,1}, z_{j,2}, \ldots, z_{j,j} = \text{root}(T_j) \) be the nodes on the path from \( u_j^i \) to \( \text{root}(T_j) \) in the tree \( T_j \). Then, for every \( i \in [r] \),

1. the first alternating-lines constraint for block \( B^1_i \) and the line \( \ell^{2i} \) is equivalent to

\[
(g_{1, w_{1, i}^{a-1}} \circ g_{1, w_{1, i}^{a-2}} \circ \cdots \circ g_{1, w_{1, i}^{1}}, \zeta_{x, \text{opt}}(p_1), \zeta_{y, \text{opt}}(\ell^{2i})) \in R_j^{1},
\]

where \( g \) is the segment reversion that reverses the whole domain of \( \ell^{2i} \).
2. if \( i > 1 \), then the second alternating-lines constraint for block \( B_i^1 \) and the line \( \ell^{2i-1} \) is equivalent to
\[
(g \circ g_{1,z_1,c_1^{i-1}} \circ g_{1,z_1,c_1^{-2}} \circ \ldots \circ g_{1,z_1,1}((\zeta^{x,\text{opt}}(p_1)), \zeta^{y,\text{opt}}(\ell^{2i}))) \in R_i^2,
\]
where \( g \) is the segment reversion that reverses the whole domain of \( p_1 \);

3. the third alternating-lines constraint for block \( B_i^2 \) and the line \( \ell^{2i} \) is equivalent to
\[
(g \circ g_{2,w_2,c_2^{i-1}} \circ g_{2,w_2,c_2^{-2}} \circ \ldots \circ g_{2,w_2,1}((\zeta^{x,\text{opt}}(p_2)), \zeta^{y,\text{opt}}(\ell^{2i}))) \in R_i^3,
\]
where \( g \) is the segment reversion that reverses the whole domain of \( p_2 \); and

4. if \( i < r \), then the fourth alternating-lines constraint for block \( B_i^3 \) and the line \( \ell^{2i+1} \) is equivalent to
\[
(g_{2,z_2,c_2^{i-1}} \circ g_{2,z_2,c_2^{-2}} \circ \ldots \circ g_{2,z_2,1}((\zeta^{x,\text{opt}}(p_2)), g((\zeta^{y,\text{opt}}(\ell^{2i+1})))) \in R_i^4,
\]
where \( g \) is the segment reversion that reverses the whole domain of \( \ell^{2i} \).

In other words, the alternating-lines constraints have segment representations of depth \( O(a) \) whose sequences of permutations correspond to root-leaf paths in two trees.

We now proceed to prove Lemma 4.10. Recall that \( p_1, p_2 \in X_0^{\text{opt}} : \ell_1, \ell_2 \in Y_{2\text{apx}}^{\text{opt}} \). We present the proof for the first two types of alternating lines constraint; the proof of the other types is analogous (i.e., one can consider a center-symmetric image of the instance with the roles of sets \( W_1 \) and \( W_2 \) swapped). That is, we show how to compute the tree \( T_1 \), the family \( G_1 \), and the relations \( (R_i^j) \) for \( j = 1, 2 \) and \( 1 \leq i \leq a \).

Let \( B_i^1(x_1), B_i^2(x_1), \ldots, B_i^a(x_1) \) be the blocks of \( W_1 \) in the bottom-to-top order in the situation \( \sigma = (p_1, p_2, \ell_1, \ell_2) \) when \( p_1 \) is positioned at \( x_1 \in D_{p_1} \). Recall that, a fixed value \( x_1 \) for \( p_1 \) determines the content of \( W_1 \cap \text{pts}_{2}(x_1, x_2) \) regardless of the choice of the value \( x_2 \) for \( p_2 \) (see Observation 4.8). Moreover, by Observation 4.9 fixing a value \( x_1 \) for \( p_1 \) also determines the partition of \( W_1 \cap \text{pts}_{1}(x_1, x_2) \) into blocks \( B_i^1(x_1) \) (which justifies the notation \( B_i^1(x_1) \)), see Figure 6.

Let \( \pi_1 : [r] \to [r] \) be a permutation such that \( B_{\pi_1(1)}^1(x_1), B_{\pi_1(2)}^1(x_1), \ldots, B_{\pi_1(r)}^1(x_1) \) is the ordering of \( B_i^1 \)'s in the decreasing order with regard to \( \leq_x \) (i.e., right-to-left) of the leaders (rightmost elements) of \( B_i^1(x_1) \). That is, we compute a permutation \( \pi_1 : [r] \to [r] \) such that for every \( x_1 \in D_{p_1} \), we have
\[
\text{leader}_{\pi_1(1)}^1(x_1) \leq_x \text{leader}_{\pi_1(2)}^1(x_1) \leq_x \ldots \leq_x \text{leader}_{\pi_1(r)}^1(x_1).
\]

Observe that \( \pi_1 \) can be computed in polynomial time using the information guessed in Branching Step E.

Similarly, let \( B_{i}^2(x_2), \ldots, B_{i}^a(x_2) \) be the blocks of \( W_2 \) in the bottom-to-top order with \( p_2 \) positioned at \( x_2 \in D_{p_2} \), and from Branching Step E we infer a permutation \( \pi_2 : [r] \to [r] \) such that for every \( x_2 \in D_{p_2} \) we have (recall that \( \text{leader}_{i}^2(x_2) \) is the leftmost element of the block \( B_i^2(x_2) \))
\[
\text{leader}_{\pi_2(1)}^2(x_2) \leq_x \text{leader}_{\pi_2(2)}^2(x_2) \leq_x \ldots \leq_x \text{leader}_{\pi_2(r)}^2(x_2).
\]

In what follows, the argument \( x_1 \) or \( x_2 \) in \( B_{i}^1(x_1) \) or \( B_{i}^2(x_2) \) will sometimes be superfluous when we only discuss the bottom-to-top order of these blocks or the left-to-right order of their leaders—these orders are fixed regardless of \( x_1 \) or \( x_2 \). In such cases we will omit the argument.

Let \( f_i : D_{p_1} \to \mathbb{N} \) be the function that assigns to \( x_1 \in D_{p_1} \) the \( y \)-coordinate of \( \text{leader}_i^1(x_1) \), \( f_i^y : D_{p_1} \to \mathbb{N} \) be the function that assigns to \( x_1 \in D_{p_1} \) the \( y \)-coordinate of the topmost element of the block \( B_i^1(x_1) \), and \( f_i^z : D_{p_1} \to \mathbb{N} \) be the function that assigns to \( x_1 \in D_{p_1} \) the \( y \)-coordinate of the bottommost element of the block \( B_i^1(x_1) \).

The main ingredient in the proof of Lemma 4.10 and our main technical result, is the following lemma, which captures the structure of possible placements of vertical lines as a tree-like application of a bounded number of segment reversions.

**Lemma 4.11.** In polynomial time, one can compute a rooted tree \( T' \) with leaves \( (T') \) = \{ \( v^1, v^2, \ldots, v_1, u^1, u^2, \ldots, u_1 \) \} and \( |V(T')}| = \( O(a) \), a family of segment reversions \( G = \{ g_v \}_{v \in V(T')} \setminus \{ \text{root}(T') \} \) , and a family of nondecreasing
functions $\tilde{F} = (\tilde{f}_v)_{v \in \text{leaves}(T')}$ such that the following holds. For every $i \in [r]$, if $v^i = v_1, v_2, \ldots, v_b = \text{root}(T')$ is the path from $v^i$ to the root $\text{root}(T')$ and $u^i = u_1, u_2, \ldots, u_c = \text{root}(T')$ is the path from $u^i$ to the root $\text{root}(T')$, then

$$f^i_1 = \tilde{f}_{v^i} \circ g_{v_b} \circ g_{v_{b-2}} \circ \ldots \circ g_{v_1},$$

$$f^i_2 = \tilde{f}_{u^i} \circ g_{u_b} \circ g_{u_{b-2}} \circ \ldots \circ g_{u_1}.$$

We now show how Lemma [4.11] implies Lemma [4.10]. First, compute the tree $T'$, segment-reversion family $G$, and family of nondecreasing functions $\tilde{F}$ via Lemma 4.11. Let $i \in [r]$ arbitrary. Note that the first alternating lines constraint is equivalent to:

$$f^i_1(\zeta^{x,\text{opt}}(p_1)) < \zeta^{y,\text{opt}}(\ell^2_i). \tag{5}$$

Let $v^i = v_1, v_2, \ldots, v_b = \text{root}(T')$ be the path from $v^i$ to the root $\text{root}(T')$. By Lemma 4.11, (5) is equivalent to

$$\tilde{f}_{v^i} \circ g_{v_b} \circ g_{v_{b-2}} \circ \ldots \circ g_{v_1}(\zeta^{x,\text{opt}}(p_1)) < \zeta^{y,\text{opt}}(\ell^2_i). \tag{6}$$

Hence, if $g$ is the segment reversion reversing $D_{\ell^2_i}$, then (6) is equivalent to

$$(\tilde{f}_{v^i} \circ g_{v_b} \circ g_{v_{b-2}} \circ \ldots \circ g_{v_1}(\zeta^{x,\text{opt}}(p_1)), g(\zeta^{y,\text{opt}}(\ell^2_i))) \in R \tag{7}$$

for some downwards-closed relation $R$. For example, we may take $R = \{(x, y) \in \mathbb{N}^2 \mid y \leq y_{\text{max}} - x\}$, where $y_{\text{max}}$ is the largest $y$-coordinate of any horizontal line. By Lemma 2.8, we can compute a downwards-closed relation $R_1^i$ such that (7) is equivalent to

$$(g_{v_b} \circ g_{v_{b-2}} \circ \ldots \circ g_{v_1}(\zeta^{x,\text{opt}}(p_1)), g(\zeta^{y,\text{opt}}(\ell^2_i))) \in R_1^i. \tag{8}$$

Similarly, if $1 < i \leq r$, the second alternating lines constraint is equivalent to

$$f^i_1(\zeta^{x,\text{opt}}(p_1)) > \zeta^{y,\text{opt}}(\ell^{2i-1}). \tag{9}$$

Let $u^i = u_1, u_2, \ldots, u_b = \text{root}(T')$ be the path from $u^i$ to the root $\text{root}(T')$. By Lemma 4.11, (9) is equivalent to

$$\tilde{f}_{u^i} \circ g_{u_b} \circ g_{u_{b-2}} \circ \ldots \circ g_{u_1}(\zeta^{x,\text{opt}}(p_1)) > \zeta^{y,\text{opt}}(\ell^{2i-1}). \tag{10}$$

Hence, if $g$ is the function reversing $\{Y_{\text{lin}}^{\text{opt}}(\ell_1), Y_{\text{lin}}^{\text{opt}}(\ell_1) + 1, \ldots, Y_{\text{lin}}^{\text{opt}}(\ell_2)\}$, then (10) is equivalent to

$$(g \circ \tilde{f}_{u^i} \circ g_{u_b} \circ g_{u_{b-2}} \circ \ldots \circ g_{u_1}(\zeta^{x,\text{opt}}(p_1)), \zeta^{y,\text{opt}}(\ell^{2i-1})) \in R \tag{11}$$

for some downwards-closed relation $R$. Define $g'$ to be the segment reversion reversing the whole $D_{\ell^2}$, and $f' = \tilde{f}_{u^i} \circ g'$. Then, since $g'$ is an involution, (11) is equivalent to:

$$(g \circ f' \circ g' \circ g_{u_b} \circ g_{u_{b-2}} \circ \ldots \circ g_{u_1}(\zeta^{x,\text{opt}}(p_1)), \zeta^{y,\text{opt}}(\ell^{2i-1})) \in R \tag{12}$$

Note that $g \circ f' = g \circ \tilde{f}_{u^i} \circ g'$ is a nondecreasing function. By Lemma 2.8 applied to $g \circ f'$ and $R$, one can compute a downwards-closed relation $R_2^i$ such that (11) is equivalent to:

$$(g' \circ g_{u_b} \circ g_{u_{b-2}} \circ \ldots \circ g_{u_1}(\zeta^{x,\text{opt}}(p_1)), \zeta^{y,\text{opt}}(\ell^{2i-1})) \in R_2^i \tag{13}$$

Using (8) and (13), the following satisfy the conditions of Lemma 4.10:

- the tree $T_1$ derived from $T'$ by adding an extra child $u'_b$ to every node $u^i$.
- the family $G$ derived from $G_1$ by adding $g_{u'_b}$, defined as the segment reversion reversing the whole $D_{\ell^2}$, and
- the relations $R_j^i$.  

27
Thus, it remains to prove Lemma 4.11.

**Proof of Lemma 4.11.** For two blocks $B'_i$ and $B'_j$, we say that a block $B'_i$ is between $B'_j$ and $B'_k$ if it is between $B'_j$ and $B'_k$ in the bottom-to-top order, that is, if $d < e$ and $d \leq j < e$ or $e < d$ and $e \leq j < d$.

Recall that $r$ is the number of blocks of $W_1$ (and of $W_2$) and recall the definition of the permutation $\pi_1$ that permutes the sequence $B'^1_1, B'^2_1, \ldots, B'^r_1$ of blocks so that their leaders are increasing in the $\leq_x$ order. We define an auxiliary rooted tree $T$ with $V(T) = [r]$ as follows. The root of $T$ is $\pi_1(1)$. For every $i \in [r] \setminus \{\pi_1(1)\}$, we define the parent of $i$ as follows. Let $i_1$ be the maximum index $i_1 < i$ with $\pi^{-1}_1(i_1) < \pi^{-1}_1(i)$ (i.e., the leader of $B'^1_i$ being to the right of the leader of $B'^1_{i_1}$). Similarly let $i_2$ be the minimum index $i_2 > i$ with $\pi^{-1}_1(i_2) < \pi^{-1}_1(i)$. These indices are undefined if the maximization or minimization is chosen over an empty set; however note that, due to the presence of $\pi^{-1}_1(1)$, at least one of these indices is defined. If exactly one is defined, we take this index to be the parent of $i$ in $T$. Otherwise, we look at the leftmost of all leaders of all blocks $B'^1_j$ between $B'^1_i$ and $B'^1_{i_1}$ (i.e., $i_1 \leq j < i$) and at the leftmost of all leaders of all blocks $B'^2_j$ between $B'^1_i$ and $B'^2_{i_2}$ (i.e., $i \leq j < i_2$) and choose as parent of $i$ the index $i_3$, $\alpha = 1, 2$, for which the aforementioned leader is more to the right (i.e., its block is later in the permutation $\pi_2$). Note that $T$ can be constructed from the information guessed in Branching Step E. See Figure 8 for an example and Figure 9 for a more involved example. In the following, the parent of a node $i$ in $T$ is denoted $\text{parent}(i)$. Furthermore, for each $i \in [r]$, we let $T_i$ be the subtree of $T$ rooted at $i$, let $B_i$ be the union of all blocks $B'_{i_1}$ for $j \in V(T_i)$.

We will use tree $T$ below to define a tree of segment partitions to which we can apply the tools from Section 2.3 yielding the required family of segment reversions. The segment partitions associated with the vertices of $T$ will be defined based on the nested behavior of blocks when moving $p_1$ in increasing $\leq_x$-order. Before we can define the partitions associated with the vertices of $T$, we need to establish a few properties of blocks.

First, no two blocks of $W_1$ share leaders.

**Claim 4.12.** Let $e \in W_1$ and assume $e$ is the leader of some $B'^1_i(x_1)$. Then, $e$ is not a leader of any block $B'^1_j(x'_1)$ with $j' \neq j$.

**Proof.** The claim follows directly from the filtering for correct orders of extremal points (Section 4.6): If $e$ is the leader of $B'^1_i(x_1)$, then $\phi(e) = \text{cell}^1_i$. (Recall that $\text{cell}^1_i$ is the cell that is expected to contain the leader of $B'^1_i(x_1)$ and is inferred from the information guessed in Branching Step E.)

Next, increasing the position of $p_1$ can only shrink blocks of $W_1$:

**Claim 4.13.** Let $x_1, x'_1$ be two elements of $D_{p_1}$ with $x_1 < x'_1$. Then for every block $B'^1_j(x'_1)$ there exists a block $B'^1_i(x_1)$ such that $B'^1_i(x'_1) \subseteq B'^1_i(x_1)$.

**Proof.** By Observation 4.6, $x'^-_2(x_1) \leq x'^-_2(x'_1)$, that is,$W_2 \cap \text{pts}_x(x_1, x'^-_2(x_1)) \subseteq W_2 \cap \text{pts}_x(x'_1, x'^-_2(x'_1))$. This immediately implies that every block $B'^1_j(x'_1)$ is contained in some block $B'^1_i(x_1)$, as desired.

Next, each leader has some well-defined interval of positions of $p_1$ during which it is the leader of its block. We first state the boundaries of this interval and then prove that they are well-defined.

Let $e \in W_1$ be the leader of some block, that is, there exist $j \in [r]$ and $x_1 \in D_{p_1}$ such that $e$ is the leader of $B'^1_j(x_1)$. Define $\text{active}^{-1}_1(e) \in D_{p_1}$ to be the maximum element of $D_{p_1}$ that is smaller than the $x$-coordinate of $e$. Define $\text{active}^{-1}_1(e)$ to be the element in $D_{p_1}$ that satisfies that $e$ is a leader of $B'^1_j(x_1)$ if and only if $\text{active}^{-1}_1(e) \leq x_1 \leq \text{active}^{-1}_1(e)$.

Note that $\text{active}^{-1}_1(e)$ is well-defined since, for $e$ to be leader of $B'^1_j(x_1)$, value $x_1 \in D_{p_1}$ needs to be smaller than the $x$-coordinate of $e$, showing that $\text{active}^{-1}_1(e)$ exists.

**Claim 4.14.** $\text{active}^{-1}_1(e)$ is well-defined.
Figure 7: Situation of Claim 4.15 and 4.16 where $p_1$ is either at position $x_1 \in D_{p_1}$ or at position $x'_1 \in D_{p_1}$ for $x < x'$. The lines of $L'_{p_1}$ for $x_1$ are denoted with $\ell^i$, $1 \leq i \leq 6$. The lines of $L'_{p_1}$ for $x'_1$ are denoted with $\ell'^i$, $1 \leq i \leq 6$. Blocks given by positioning $p_1$ at $x'_1$ ($B_1^1(x'_1), B_2^1(x'_1), B_3^1(x'_1)$) are depicted by purple color with circled leaders and blocks given by positioning $p_1$ at $x_1$ ($B_1^1(x_1), B_2^1(x_1), B_3^1(x_1)$) are depicted by the union of green and purple color with squared leaders.

Proof. It suffices to show that, if $e$ is the leader of $B_1^1(x_1)$ for some $x_1 \in D_{p_1}$, then it is also the leader of $B_1^1(x'_1)$ for every $x'_1 \in D_{p_1}$ with $x_1 \leq x'_1 \leq \text{active}^{-1}(e)$ (note that $x_1 \leq \text{active}^{-1}(e)$ by the definition of $\text{active}^{-1}(e)$).

Let $B_1^1(x'_1)$ be the block containing $e$ and let $e'$ be the leader of this block. By Claim 4.13 and the fact that $B_1^1(x'_1)$ and $B_1^1(x_1)$ share $e$ we have $B_1^1(x'_1) \subseteq B_1^1(x_1)$. Thus, the fact that $e'$ is the leader of $B_1^1(x'_1)$ implies $e \leq x e'$ while the fact that $e$ is the leader of $B_1^1(x_1)$ implies $e' \leq x e$. Hence, $e = e'$. By Claim 4.12 $j = j'$ and we are done.

Intuitively, there are two things that can happen to a block with some index $j$ when moving $p_1$ to the right: It can shrink, or it can disappear and reappear elsewhere. Now, if increasing the position of $p_1$ shrinks a block but it does not disappear, then the leader stays the same:

Claim 4.15. If for some $x_1, x'_1 \in D_{p_1}$ with $x_1 < x'_1$ and an index $j \in [r]$ we have $B_j^1(x'_1) \subseteq B_j^1(x_1)$, then $\text{leader}_j^1(x_1) = \text{leader}_j^1(x'_1)$.

Proof. Since $B_j^1(x'_1) \subseteq B_j^1(x_1)$, the leader $\text{leader}_j^1(x_1)$ is to the right of the coordinate $x'_1$. Thus, $x'_1 \leq \text{active}^{-1}(\text{leader}^1_j(x_1))$ by the definition of $\text{active}^{-1}$. Thus, $\text{leader}_j^1(x_1)$ is also a leader of $B_j^1(x'_1)$.

Next we observe that, when moving $p_1$ to the right from one position $x_1$ to another position $x'_1$, then, when ordering the blocks of $W_1$ according to $\pi_1$, that is, increasing $x$-coordinates of leaders, then there is a unique block index $i_{x_1 \leftarrow x'_1}$ such that blocks before $i_{x_1 \leftarrow x'_1}$ disappear and reappear elsewhere, and blocks after $i_{x_1 \leftarrow x'_1}$ may shrink but do not disappear.

Let $x_1, x'_1$ be two elements of $D_{p_1}$ with $x_1 < x'_1$. Define $i_{x_1 \leftarrow x'_1}$ as the unique index $i_{x_1 \leftarrow x'_1} \in [r]$ that satisfies that, for every $j \in [r]$, we have $B_j^1(x'_1) \subseteq B_j^1(x_1)$ if and only if $\pi_1^{-1}(j) \leq i_{x_1 \leftarrow x'_1}$.

Claim 4.16. Index $i_{x_1 \leftarrow x'_1}$ is well-defined.
Proof. Let $j \in [r]$ be such that $B^j_1(x'_1) \subseteq B^j_1(x_1)$ for some $j' \neq j$ and let $j'' \in [r]$ be such that $\pi^{-1}_1(j) < \pi^{-1}_1(j'')$. Note that $B^j_1(x'_1) \cap B^j_1(x_1) = \emptyset$ and that it suffices to show that also $B^{j''}_1(x'_1) \cap B^{j''}_1(x_1) = \emptyset$.

Since $B^j_1(x'_1) \subseteq B^j_1(x_1)$, by Claim 4.12 active$\rightarrow$ (leader$^{-1}_1(x_1)) < x'_1$. By definition of active$\rightarrow$ and the discretization properties thus leader$^{-1}_1(x_1)$ is to the left of $x'_1$. Since $\pi^{-1}_1(j) < \pi^{-1}_1(j'')$, we have leader$^{-1}_1(x_1) \leq x_1$ leader$^{-1}_1(x_1)$. Thus active$\rightarrow$ (leader$^{-1}_1(x_1)) < x'_1$. Hence, $B^{j''}_1(x'_1) \cap B^{j''}_1(x_1) = \emptyset$ as desired.

For every $j \in [r]$ and two elements $x_1, x'_1 \in D_{p_1}$ with $x_1 < x'_1$ we define $\alpha_{x_1\leftarrow x'_1}(j) \in [r]$ as follows. Let $\alpha_{x_1\leftarrow x'_1}(j)$ be the ancestor of $j$ in $T$ that is closest to $j$ in $T$ such that for at least one block $B^j_2$ between $B^j_1(x'_1)$ and $B^j_1$ the leader of $B^j_2(x'_2(x'_1))$ is to the left of the $x$-coordinate $x''_1(x_1)$. We put $\alpha_{x_1\leftarrow x'_1}(j)$ to be the root $\pi(1)$ if such an ancestor does not exist.

The intuition behind the notion $\alpha_{x_1\leftarrow x'_1}(j)$ is the following: If we slide $p_1$ from $x_1$ to the right to $x'_1$, then the $j$-th block $B^j_1$ at $x'_1$ is a subset of $B^{\alpha_{x_1\leftarrow x'_1}(j)}_1$ at $x_1$. Furthermore, for every descendant $j_1$ of $j$ such that $B^{j_1}_1$ is a subset of $B^{\alpha_{x_1\leftarrow x'_1}(j)}_1$. We now prove this intuition in the next three claims. We start with the following intermediate step.

Claim 4.17. Let $x_1, x'_1$ be two elements of $D_{p_1}$ with $x_1 < x'_1$ and let $j \in [r]$. Then

$$B^j_1(x'_1) \cup B^{\alpha_{x_1\leftarrow x'_1}(j)}_1(x'_1) \subseteq B^{\alpha_{x_1\leftarrow x'_1}(j)}_1(x_1).$$

Proof: Let $j = j_1, j_2, \ldots, j_b = \alpha_{x_1\leftarrow x'_1}(j)$ be the vertices on the path in $T$ from $j$ to $\alpha_{x_1\leftarrow x'_1}(j)$. By the definition of $\alpha_{x_1\leftarrow x'_1}(j)$, for every $i \in [b - 1]$, the leaders of all blocks $B^j_{i+1}(x'_1)$ are to the right of $x''_1(x_1)$, that is, the blocks $B^j_{i+1}(x''_1(x_1))$ are disjoint with $\text{pts}_x(x_1, x''_1(x_1))$. Thus, there is no point of $W_2$ in the area bounded by $x''_1(x_1)$, the predecessor $p_2$ of $p_2$ in $X_{im}$, and the two lines given by the $y$-coordinates of the topmost and bottommost point, respectively, in the blocks $B^j_{i+1}(x_1)$. This implies that all blocks $B^j_{i+1}(x'_1), i \in [b],$ are contained in the same block $B^{j_1}_1(x_1)$.

We now show that $\alpha_{x_1\leftarrow x'_1}(j)$ is the first index $j'$ in the sequence $\pi(1), \pi(2), \ldots, \pi(r)$ such that $B^{j'}_{i+1}(x'_1)$ is a subset of $B^{\alpha_{x_1\leftarrow x'_1}(j)}_1(x_1)$. The claim is immediate if $\alpha_{x_1\leftarrow x'_1}(j) = \pi(1)$, so assume otherwise. Then, $\alpha_{x_1\leftarrow x'_1}(j)$ is not the root of $T$ and thus parent$(\alpha_{x_1\leftarrow x'_1}(j))$ is defined. Assume that there exists an index $j_0$ with $\pi^{-1}_1(j_0) < \pi^{-1}_1(x_{1\leftarrow x'_1}(j))$ such that $B^{j_0_1}(x'_1) \subseteq B^{j_0}_1(x_1)$. This implies that for every $j_1$ between $B^{\alpha_{x_1\leftarrow x'_1}(j)}_1$ and $B^{j_0}_1$, the leader of $B^{j_1}_1(x'_1)$ is to the left of $x''_1(x_1)$. If $j_0 = \text{parent}(\alpha_{x_1\leftarrow x'_1}(j))$, then this is a contradiction to the fact that, by definition of $\alpha_{x_1\leftarrow x'_1}(j)$, there is a block $B^{j_0}_1(x'_1)$ between $B^{j_0}_1$ and $B^{\alpha_{x_1\leftarrow x'_1}(j)}_1$ whose leader is to the right of $x''_1(x_1)$. If $j_0 \neq \text{parent}(\alpha_{x_1\leftarrow x'_1}(j))$, then it follows that the leftmost of the leaders of blocks of $W_2$ between $B^{j_0}_1$ and $B^{\alpha_{x_1\leftarrow x'_1}(j)}_1$ is more to the right than the leftmost of the leaders of blocks of $B_2$ between $B^{\alpha_{x_1\leftarrow x'_1}(j)}_1$ and $B^{\text{parent}(\alpha_{x_1\leftarrow x'_1}(j))}_1$. This is a contradiction to the definition of parents. Thus indeed $\alpha_{x_1\leftarrow x'_1}(j)$ is the earliest index $j'$ in the sequence $\pi(1), \pi(2), \ldots, \pi(r)$ such that $B^{j'}_1(x'_1)$ is a subset of $B^{j_0}_1(x_1)$.

We conclude that the leader (rightmost element) $e$ of $B^{j_0}_1(x_1)$ is the leader (rightmost element) of $B^{\alpha_{x_1\leftarrow x'_1}(j)}_1(x'_1)$; by Claim 4.12 it implies that $\alpha_{x_1\leftarrow x'_1}(j) = j_0$. This establishes (14).

In the next claim, we treat blocks that only shrink (but do not disappear) when $p_1$ slides from $x_1$ to $x'_1$.

Claim 4.18. Let $x_1, x'_1$ be two elements of $D_{p_1}$ with $x_1 < x'_1$ and let $j \in [r]$. Then the following conditions are equivalent:

$\footnote{That is, this ancestor has the shortest path to $j$ in $T$. A node is an ancestor of itself, that is, it can happen that $\alpha_{x_1\leftarrow x'_1}(j) = j$.}$

$\footnote{Node $j$ is its own descendant.}$
Figure 8: Situation of Claim 4.19, where $p_1$ is either at position $x_1 \in D_{p_1}$ or at position $x'_1 \in D_{p_1}$ for $x < x'$. The lines of $L'_i$ for $x_1$ are denoted with $\ell_i$, $1 \leq i \leq 6$. The lines of $L'_i$ for $x'_1$ are denoted with $\ell'_i$, $1 \leq i \leq 6$. Blocks given by positioning $p_1$ at $x'_1$ ($B'_1(x'_1)$) are depicted by purple color with circled leaders and blocks given by positioning $p_1$ at $x_1$ ($B'_1(x_1)$) are depicted by the union of green and purple color with squared leaders. Blocks given by positioning $p_2$ at $x'_2$ ($x'_2$) with circled leaders. Observe that $B'_1(x'_1) \not\subseteq B'_1(x_1)$ and $\alpha_{x_1 \leftarrow x'_1}(3) = 2$. Then $B'_1(x'_1) \cup B'_1(x'_2) \subseteq B'_1(x_1)$.

1. $B'_1(x'_1) \subseteq B'_1(x_1)$;
2. $\text{leader}_1(x_1) = \text{leader}_1(x'_1)$;
3. $\alpha_{x_1 \leftarrow x'_1}(j) = j$.

Proof. If $B'_1(x'_1) \subseteq B'_1(x_1)$, then $\text{leader}_1(x_1) = \text{leader}_1(x'_1)$ by Claim 4.15. In the other direction, if $\text{leader}_1(x_1) = \text{leader}_1(x'_1)$, then $B'_1(x'_1) \cap B'_1(x_1) \neq \emptyset$, so Claim 4.13 implies $B'_1(x'_1) \subseteq B'_1(x_1)$.

To prove equivalence of the first and third condition, we use (14) of Claim 4.17 which implies that $B'_1(x'_1) \subseteq B'_{x_1 \leftarrow x'_1}(j)(x_1)$. Since the blocks $B'_1(x_1)$ are disjoint for distinct $i \in [r]$, $B'_1(x'_1) \subseteq B'_1(x_1)$ is equivalent to $\alpha_{x_1 \leftarrow x'_1}(j) = j$.

In the last claim we show that, when moving $p_1$ from $x_1$ to $x'_1$, and it is the case that the $j$th block $B'_1$ disappears and reappears elsewhere, then $B'_1(x'_1)$ and all the blocks at position $x'_1$ corresponding to descendants of $j$ in $T$ are contained in $B'_{x_1 \leftarrow x'_1}(j)(x_1)$. The formal statement is as follows.

Claim 4.19. Let $x_1, x'_1$ be two elements of $D_{p_1}$ with $x_1 < x'_1$ and let $j \in [r]$ be such that $B'_1(x'_1) \not\subseteq B'_1(x_1)$. Then for every descendant $j_1$ of $j$ in $T$ we have $B'_{j_1}(x'_1) \subseteq B'_{x_1 \leftarrow x'_1}(j)(x_1)$.

Proof. Since $B'_1(x'_1) \not\subseteq B'_1(x_1)$, we have $\pi_1^{-1}(j) > i_{x_1 \leftarrow x'_1}$. For every $j_1 \in V(T_j) \setminus \{j\}$ we have $\pi_1^{-1}(j_1) > \pi_1^{-1}(j)$ and thus $\pi_1^{-1}(j_1) > i_{x_1 \leftarrow x'_1}$ as well. In particular, $\text{leader}_1(x_1)$ is to the left of $x'_1$ and hence $B'_{j_1}(x_1) \cap B'_1(x_1) = \emptyset$. Hence, no vertex $j'$ on the path in $T$ between $j_1$ and $j$ (including $j_1$ and $j$) satisfies $B'_{j_1}(x'_1) \subseteq B'_1(x_1)$. Thus, applying (14) in Claim 4.17 to $j_1$ (instead of $j$), we obtain that $\alpha_{x_1 \leftarrow x'_1}(j_1) \neq j'$. Thus, $\alpha_{x_1 \leftarrow x'_1}(j_1)$ is an ancestor of $j$. By definition of $\alpha_{x_1 \leftarrow x'_1}(j)$ we conclude that $\alpha_{x_1 \leftarrow x'_1}(j_1) = \alpha_{x_1 \leftarrow x'_1}(j)$. Now
Figure 9: More complex example of situation in Claim 4.19 where \( p_1 \) is either at position \( x_1 \in D_{p_1} \) or at position \( x'_1 \in D_{p_1} \) for \( x < x' \). The lines of \( \tilde{L}_\sigma' \) for \( x_1 \) are denoted with dashed and dotted lines. The lines of \( \tilde{L}_\sigma' \) for \( x'_1 \) are denoted with dashed and dash-dotted lines. Blocks given by positioning \( p_1 \) at \( x_1 \) are depicted by purple color with circled leaders and blocks given by positioning \( p_1 \) at \( x_1 \) are depicted by the union of green and purple color with squared leaders. Blocks given by positioning \( p_2 \) at \( x_1 \) are depicted by orange color with squared leaders and blocks given by positioning \( p_1 \) at \( x'_1 \) are depicted by the union of yellow and orange color with circled leaders. An auxiliary rooted tree \( T \) for red blocks is also visualized.
we may use the notation $T_j$, and the whole block $B_j^i$ is monotonous, that is, if $x_1 < \text{active}_j^i(e)$, the element $e$ and the whole block $B_j^i(\text{active}_j^i(e))$ is a subset of some other block $B_j^i(x_1)$ for an ancestor $j'$ of $j$ in the tree $T$. Furthermore, $B_j^i(\text{active}_j^i(e))$ is also a subset of $B_j^i(x_1)$ for every $j_i \in V(T_j)$. For a block $B_j^i$ and an element $e$ that is the leader of $B_j^i(x_1)$ for some $x_1 \in D_p$, the epoch of $B_j^i$ and $e$ is the interval $[\text{active}_j^i(e), \text{active}_j^i(e)]$ in $D_p$. Note that each block $B_j^i$ partitions $D_p$ into epochs; let $\mathcal{P}_j$ be this partition. Note that the epochs one-to-one correspond to the intervals $[\text{active}_j^i(e), \text{active}_j^i(e)]$ where $e$ is the leader of some block in $W_1$ for some $x_1$. Moreover, $e$ is unique to this interval. Hence, for an epoch $e$, we may use the notation $e = [\text{active}_j^i(e), \text{active}_j^i(e)]$ without ambiguity.

We now make several observations about the structure of epochs. Let $x_1, x_1' \in D_p$ with $x_1 < x_1'$. Claims 4.14 and 4.15 ensure that if $x_1, x_1'$ belong to different epochs of $B_j^i$, then $B_j^i(x_1) \cap B_j^i(x_1') = \emptyset$, and if $x_1, x_1'$ belong to the same epoch of $B_j^i$, then $B_j^i(x_1) \subseteq B_j^i(x_1')$ and $\text{leader}_j^i(x_1) = \text{leader}_j^i(x_1')$. Claims 4.18 and 4.19 ensure that, if $j'$ is an ancestor of $j$ in $T$, then the epochs of $B_j^i$ are supersets of the epochs of $B_{j'}^i$, that is, the epochs of $B_j^i$ form a coarser partition of $D_p$ into segments than the epochs of $B_{j'}^i$. To see this, consider two distinct epochs $e, e'$ of $B_j^i$ where $e$ is to the left of $e'$ and observe that the leader of $B_j^i$ is different in these two epochs. It then suffices to show that also the leader of $B_j^i(x_1), x_1 \in e_i$, is different from the leader of $B_{j'}^i(x_1'), x_1' \in e_2$. By Claim 4.18 we have $B_j^i(x_1') \subseteq B_j^i(x_1)$, and by Claim 4.19 thus $B_j^i(x_1) \subseteq B_j^i(x_1')$, that is, $B_j^i(x_1') \subseteq B_j^i(x_1)$. By Claim 4.18 the leader $B_j^i(x_1)$ is different from $B_j^i(x_1')$. Finally, observe also that $B_j^i(x_1)$ has only one epoch, because $\text{leader}_j^i(x_1)$ is the rightmost point of $\text{apocell}(j_i, e_1)$ and thus stays constant for all $x_1 \in D_p$.

For an epoch $e$ of a block $B_j^i$, we denote by $\text{active}_j^i(e)$ and $\text{active}_j^i(e)$ the minimum and maximum $y$-coordinate of an element of $B_j^i(x_1)$ for $x_1 \in e$ and $j_i \in V(T_j)$. Note that the minimum and maximum values are always attained for $x_1 = \text{active}_j^i(e)$, as the union of all blocks $B_j^i(x_1)$ for $j_i \in V(T_j)$ only grows (in the subset order) as $x_1$ decreases from $\text{active}_j^i(e)$ to $\text{active}_j^i(e)$.

By definition, if $j'$ is an ancestor of $j$ in $T$ and $e'$ is an epoch of $B_j^i$ that contains an epoch $e$ of $j$, then

$$\text{active}_j^i(e), \text{active}_j^i(e) \subseteq [\text{active}_j^i(e'), \text{active}_j^i(e')].$$

Thus, with an epoch $e$ one can associate a rectangle in $\mathbb{R}^2$:

$$[\text{active}_j^i(e), \text{active}_j^i(e)] \times [\text{active}_j^i(e), \text{active}_j^i(e)],$$

and we have that the rectangle of an epoch of a block $B_j^i$ is contained in the rectangle of a corresponding epoch of a block $B_{j'}^i$ for an ancestor $j'$ of $j$.

Claim 4.19 implies that, for two different epochs $e$ and $e'$ of the same block $B_j^i$, the segments $[\text{active}_j^i(e), \text{active}_j^i(e)]$ and $[\text{active}_j^i(e'), \text{active}_j^i(e')]$ are disjoint. (Note that, since the leader of $B_j^i$ changes between the two epochs, Claim 4.19 implies that the leader of $B_j^i$ changes for each descendant $j_i$ of $j$, that is, the corresponding blocks are disjoint.) We now observe that, moreover, for a right-to-left sequence of epochs of some block, these segments are ordered top-to-bottom or vice versa (see also Figure 10):

Claim 4.20. Let $j'$ be the parent of $j$ in $T$, let $e'$ be an epoch of $j'$, and let $e_1, e_2, \ldots, e_b$ be the epochs of $j$ contained in $e'$ in the right-to-left order. Then the sequence of disjoint segments $[\text{active}_j^i(e), \text{active}_j^i(e_i)]_{i=1}^n$ is monotonous, that is, if $j < j'$ then

$$\text{active}_j^i(e_1) \geq \text{active}_j^i(e_2) \geq \ldots \geq \text{active}_j^i(e_b).$$
Figure 10: Statement of Claim 4.20. The big rectangle is the box \([\text{active}^+_{B_1}(\varepsilon), \text{active}^-_{B_1}(\varepsilon)] \times \text{active}^+_{B_1}(\varepsilon), \text{active}^-_{B_1}(\varepsilon)]\) for one epoch of a block \(B_1^j\) with two children \(j_1 > j\) and \(j_2 < j\). The small boxes above with north east lines correspond to epochs of \(B_1^{j_1}\) and the small boxes below with north west lines correspond to epochs of \(B_1^{j_2}\). The horizontal dashed lines indicate the \(y\)-coordinate of the leader at the corresponding epoch.

and if \(j' < j\) then

\[
\text{active}^+_{B_1}(\varepsilon_1) \leq \text{active}^+_{B_1}(\varepsilon_1) < \text{active}^+_{B_1}(\varepsilon_2) \leq \text{active}^+_{B_1}(\varepsilon_2) < \ldots < \text{active}^+_{B_1}(\varepsilon_b) \leq \text{active}^+_{B_1}(\varepsilon_b).
\]

**Proof.** Recall that through the entire epoch \(\varepsilon\) the leader of the block \(B_1^j\) stays the same: if \(x_1 < x'_1\) for \(x_1, x'_1 \in \varepsilon\) then \(\text{leader}^j_1(x_1) = \text{leader}^j_1(x'_1)\) and \(B_1^j(x_1) \subseteq B_1^j(x'_1)\). Claim 4.17 implies moreover that \(\alpha_{x_1,x'_1}(j') = j'\).

Fix \(\beta \in [b - 1]\), \(x'_1 \in \varepsilon_{\beta}\), and \(x_1 \in \varepsilon_{\beta + 1}\). We have \(B_1^j(x'_1) \not\subseteq B_1^j(x_1)\). Claim 4.18 implies that \(j \neq \alpha_{x_1,x'_1}(j)\). By Claim 4.19 applied to \(B_1^j, x_1, x'_1\) we infer that \(\alpha_{x_1,x'_1}(j) = j'\) as \(j'\) is the parent of \(j\) and \(B_1^j(x'_1) \subseteq B_1^j(x_1)\). Furthermore, we have \(B_1^j(x'_1) \subseteq B_1^j(x_1)\) for every \(j_1 \in V(T_j)\). On the other hand, \(B_1^j(x_1)\) is below \(B_1^j(x_1)\) if \(j < j'\) and above \(B_1^j(x_1)\) if \(j > j'\). Hence, \(\text{active}^+_{B_1}(\varepsilon_{\beta+1}) < \text{active}^+_{B_1}(\varepsilon_\beta)\) if \(j < j'\) and \(\text{active}^+_{B_1}(\varepsilon_{\beta+1}) > \text{active}^+_{B_1}(\varepsilon_\beta)\) if \(j > j'\). This finishes the proof of the claim. 

Claim 4.20 allows us to conclude the proof of Lemma 4.11 as follows using the setting of Section 2.3.

We construct a tree \(T'\) from \(T\) by appending to every \(j \in V(T) = [r]\) two new children \(v'\) and \(w'\) (which are leaves of \(T'\)). With every node \(j \in [r]\) we associate the segment partition \(\mathcal{P}_j\) of \(D_{p_1}\) into epochs of \(B_1^j\) and with every leaf of \(T'\) we associate the most refined segment partition of \(D_{p_1}\) with only singletons. For every non-root node \(j \in V(T)\) we define \(\text{type}(j) = \text{inc}\) if \(j < \text{parent}(j)\) and \(\text{type}(j) = \text{dec}\) if \(j > \text{parent}(j)\). We also define \(\text{type}(v') = \text{dec}\) and \(\text{type}(w') = \text{inc}\). This makes \(T = ((D_{p_1}, \leq), T', (\mathcal{P}_v)_{v \in V(T')}, \text{type})\) a tree of segment partitions.

Now define for every \(j \in [r]\) functions \(f_{v'} = f^j_1\) and \(f_{w'} = f^j_2\). Observe that within one epoch of \(B_1^j, f^j_1\) is nonincreasing and \(f^j_2\) is nondecreasing. Consequently, Claim 4.20 (together with the fact that \(B_1^{\pi(1)}\) has only one epoch) implies that the family of functions \(\mathcal{F} = (f_v)_{v \in \text{leaves}(T')}\) is a family of leaf functions for the tree of segment partitions \(T\).
We apply Lemma 2.9 to $T$ and $F$ and obtain a family $G = (g_v)_{v \in V(T') \setminus \text{root}(T)}$ of segment reversions and a family $\tilde{F} = (\tilde{f}_v)_{v \in \text{leaves}(T)}$ of nondecreasing functions. By Lemma 2.9, we can return $T'$, $G$, and $\tilde{F}$ as outcomes of Lemma 4.11.

### Figure 11: Illustration of the proof of Lemma 4.21

Solid lines are from $X_{\text{lin}}^{\text{apx}} \cup Y_{\text{lin}}^{\text{apx}}$, dashed lines are from $X_{\text{lin}}^{\text{opt}} \cup Y_{\text{lin}}^{\text{opt}}$. The apx-supercells $\text{apxcell}(p_1', \ell_1)$ and $\text{apxcell}(p_2', \ell_1')$ and the opt-supercell $\text{optcell}(p_1, q_1)$ are highlighted.

#### 4.8 Completeness

We now perform a tedious but rather direct check that shows that all defined constraints and steps where we filtered out domains of some lines guarantee completeness.

**Lemma 4.21.** If an assignment $\zeta^{x, \text{opt}}(\zeta^{y, \text{opt}})$ that assigns to every line $\ell$ an element in $D_\ell$ satisfies all monotonicity, corner, alternation, and alternating lines constraints, then the pair $((\zeta^{x, \text{opt}}(\ell) \mid \ell \in X_{\text{lin}}^{\text{opt}}), (\zeta^{y, \text{opt}}(\ell) \mid \ell \in Y_{\text{lin}}^{\text{opt}})$ is a separation.

**Proof.** The proof is by contradiction. Assume that there exist two points $(x_1, y_1) \in W_1$ and $(x_2, y_2) \in W_2$ such that no element of $X' := \{\zeta^{x, \text{opt}}(\ell) \mid \ell \in X_{\text{lin}}^{\text{opt}}\}$ is between $x_1$ and $x_2$ and no element of $Y' := \{\zeta^{y, \text{opt}}(\ell) \mid \ell \in Y_{\text{lin}}^{\text{opt}}\}$ is between $y_1$ and $y_2$. Our goal is to obtain a contradiction by exhibiting either a violated constraint or an element $\zeta^{x, \text{opt}}(\ell)$ or $\zeta^{y, \text{opt}}(\ell)$ of some domain $D_\ell$ that should have been removed in one of the filtering steps.

Let $\zeta^x = \zeta^{x, \text{opt}} \cup \zeta^{x, \text{apx}}$ and $\zeta^y = \zeta^{y, \text{opt}} \cup \zeta^{y, \text{apx}}$. Observe that the choice of the domains and the monotonicity constraints ensure that $\zeta^x$ and $\zeta^y$ are both increasing functions.

Let $p_1 \in X_{\text{lin}}$ be such that $\zeta^x(p_1)$ is the maximum element of $X' \cup \{1\}$ that is smaller than $x_1$ and $x_2$ and let $p_2 \in Y_{\text{lin}}$ be such that $\zeta^x(p_2)$ is the successor of $\zeta^x(p_1)$ in $X' \cup \{1, 3n + 1\}$. Note that $\zeta^x(p_2) > x_1, x_2$. Similarly, let $q_1 \in Y_{\text{lin}}$ be such that $\zeta^y(q_1)$ be the maximum element of $Y' \cup \{1\}$ that is smaller than $y_1$ and $y_2$ and let $q_2 \in Y_{\text{lin}}$ be such that $\zeta^y(q_2)$ be the successor of $\zeta^y(q_1)$ in $Y' \cup \{1, 3n + 1\}$. Again, $\zeta^y(q_2) > y_1, y_2$.

By symmetry, we can assume that $\delta(\text{optcell}(p_1, q_1)) \neq 1$. Let $p_1' \in X_{\text{lin}}^{\text{apx}}$ and $\ell_1 \in Y_{\text{lin}}^{\text{apx}}$ such that $\text{apxcell}(p_1', \ell_1)$ is the apx-supercell containing $(x_1, y_1)$. Note that $\delta(\text{apxcell}(p_1', \ell_1)) = 1$. Let $p_1'$ be the successor...
of \( p'_1 \) in \( X_{\text{lin}}^{\text{apx}} \) and let \( \ell_2 \) be the successor of \( \ell_1 \) in \( Y_{\text{lin}}^{\text{apx}} \). Let \( \hat{p}_1 \) be the maximum of the pair \( \{ p'_1, p_1 \} \), \( \hat{\ell}_1 \) be the maximum of the pair \( \{ p'_2, \ell_2 \} \), \( \hat{\ell}_1 \) be the maximum of the pair \( \{ q_1, \ell_1 \} \), and \( \hat{\ell}_2 \) be the minimum of the pair \( \{ q_2, \ell_2 \} \). Note that \( \zeta^- (\hat{p}_1) < x_1 < \zeta^- (\hat{p}_2) \) and \( \zeta^- (\hat{\ell}_1) < y_1 < \zeta^- (\hat{\ell}_2) \). Consult Figure 11 for an example of such situation.

**Claim 4.22.** Exactly one of the lines \( \hat{p}_1, \hat{p}_2, \hat{\ell}_1, \) and \( \hat{\ell}_2 \) lies in \( X_{\text{lin}}^{\text{apx}} \cup Y_{\text{lin}}^{\text{apx}} \).

**Proof.** First, we exclude the case when \( \hat{p}_1 = p_1, \hat{p}_2 = p_2, \hat{\ell}_1 = q_1, \) and \( \hat{\ell}_2 = q_2 \). If this were the case, then both \((x_1, y_1)\) and \((x_2, y_2)\) would lie in the \( \text{apx-supercell} \) \( \text{apxcell}(p'_1, \ell_1) \), contradicting the fact that \((X_0, Y_0)\) is a separation. (Recall that \((X_0, Y_0)\) is the initially computed 2-approximate separation.)

Consider now the case when at least two of the lines \( \hat{p}_1, \hat{p}_2, \hat{\ell}_1, \hat{\ell}_2 \) belong to \( X_{\text{lin}}^{\text{apx}} \cup Y_{\text{lin}}^{\text{apx}} \). Then the area of interest of the tuple \((\hat{p}_1, \hat{p}_2, \hat{\ell}_1, \hat{\ell}_2)\) lies inside the \( \text{apx-supercell} \) \( \text{apxcell}(p'_1, \ell_1) \) and also inside the \( \text{opt-supercell} \) \( \text{optcell}(p_1, q_1) \). Consider the abstract cell \( C \) corresponding to \((\hat{p}_1, \hat{p}_2, \hat{\ell}_1, \hat{\ell}_2)\). Since \( \delta(\text{apxcell}(p'_1, \ell_1)) = 1 \), we have \( \delta(C) \in \{0, 1\} \) by definition of \( \delta \). Since \( \delta(\text{optcell}(p_1, q_1)) \neq 1 \) furthermore \( \delta(C) = 0 \) (again, by definition of \( \delta \)). Thus, the tuple \((\hat{p}_1, \hat{p}_2, \hat{\ell}_1, \hat{\ell}_2)\) is an empty corner. However, then the presence of \((x_1, y_1)\) in the area of interest of the tuple \((\hat{p}_1, \hat{p}_2, \hat{\ell}_1, \hat{\ell}_2)\) is a contradiction as it violates the corner constraint or the corner filtering step for the empty corner in question.

Thus, exactly one of the lines \( \hat{p}_1, \hat{p}_2, \hat{\ell}_1, \) and \( \hat{\ell}_2 \) lies in \( X_{\text{lin}}^{\text{apx}} \cup Y_{\text{lin}}^{\text{apx}} \). We claim that, by symmetry, we may assume without loss of generality that this is \( p_2 \). That is, \( \hat{p}_1 = p_1, \hat{p}_2 = p'_2, \hat{\ell}_1 = q_1, \) and \( \hat{\ell}_2 = q_2 \). In other words, \( p'_1 < p_1 < p'_2 < p_2 \) and \( \ell_1 < q_1 < q_2 < \ell_2 \). Indeed, to see that the above symmetry-breaking assumption is without loss of generality, we may use the fact that the addition of alternation constraints and alternating lines constraints as well as filtering of correct orders of extremal points has been performed both in top/down and left/right directions, and that in the following we will solely rely on these filtering steps and constraints. Observe that now \( \sigma = (p_1, p_2, \ell_1, \ell_2) \) is a situation.

Let \( p'_2 \) be the maximum element of \( X_{\text{lin}}^{\text{apx}} \) that is smaller than \( p_2 \). Recall that \( L_\sigma \) is the set of lines of \( Y_{\text{lin}}^{\text{opt}} \) between \( \ell_1 \) and \( \ell_2 \), \( L'_\sigma = L_\sigma \cup \{ \ell_1 \} \), and \( \text{area}(\ell) = \text{area}(p_1, p_2, \ell, \ell') \) for every \( \ell \in L'_\sigma \) where \( \ell' \) is the successor of \( \ell \) in \( L_\sigma \cup \{ \ell_1, \ell_2 \} \).

**Claim 4.23.** There exists an element \( \ell \in Y_{\text{lin}}^{\text{opt}} \), \( q_2 \leq \ell < q_2 \), such that \( \delta(\text{optcell}(p_1, \ell)) = 1 \). Similarly, there exists \( \ell \in Y_{\text{lin}}^{\text{opt}} \) with \( \ell \leq q_1 \) such that \( \delta(\text{optcell}(p_1, \ell)) = 1 \).

**Proof.** We show only the first claim, the proof for the second one is analogous. Assume the contrary. Then, as \( \delta(\text{apxcell}(p'_1, \ell_1)) = 1 \) while for every \( \ell \in Y_{\text{lin}}^{\text{opt}} \) with \( q_1 \leq \ell < q_2 \) we have \( \delta(\text{optcell}(p_1, \ell)) \neq 1 \), every cell in the area of interest of the tuple \((p_1, p'_2, q_1, \ell_2)\) satisfies \( \delta(\text{cell}) = 0 \). Hence, the tuple \((p_1, p'_2, q_1, \ell_2)\) is an empty corner. However, the existence of \((x_1, y_1)\) violates the corner filtering or the corner constraint for that tuple.

**Claim 4.24.** Assume \( \delta(\text{optcell}(p_1, q_1)) = 0 \). Then, \((x_2, y_2)\) lies in the \( \text{apx-supercell} \) \( \text{apxcell}(p'_2, \ell_1) \) and, consequently, \( \delta(\text{apxcell}(p'_2, \ell_1)) = 2 \). Furthermore, there exists \( \ell \in Y_{\text{lin}}^{\text{opt}} \) with \( q_1 \leq \ell < q_2 \) such that \( \delta(\text{optcell}(p_1, \ell)) = 2 \).

**Proof.** Recall that \((x_2, y_2)\) lies in the \( \text{opt-supercell} \) \( \text{optcell}(p_1, q_1) \). Since \( p'_1 < p_1 < p'_2 < p_2 \) and \( \ell_1 < q_1 < q_2 < \ell_2 \), the \( \text{apx-supercells} \) that share cells with \( \text{optcell}(p_1, q_1) \) are the cells \( \text{apxcell}(r, \ell_1) \) for \( p'_1 \leq r \leq p'_2 \). Assume \((x_2, y_2)\) lies in \( \text{apxcell}(r, \ell_1) \) for some \( p'_1 \leq r \leq p'_2 \).

Since \((x_2, y_2)\) lies in \( W_2 \), point \((x_2, y_2)\) does not lie in the \( \text{apx-supercell} \) \( \text{apxcell}(p'_1, \ell_1) \), so \( r \neq p'_1 \). If \( r < p'_2 \), then consider the tuple \((r, r', q_1, q_2)\) where \( r' \) is the successor of \( r \) in \( X_{\text{lin}}^{\text{apx}} \). Observe that the area of interest of that tuple is contained in the \( \text{apx-supercell}(r, \ell_1) \) and in the \( \text{opt-supercell} \) \( \text{optcell}(p_1, q_1) \) and contains \((x_2, y_2)\). Since \((x_2, y_2)\) is in that \( \text{apx-supercell} \), \( \delta(\text{apxcell}(r, \ell_1)) = 2 \). Since \( \delta(\text{optcell}(p_1, q_1)) = 0 \), for every cell \( r \) in the area of interest of \((r, r', q_1, q_2)\) we have \( \delta(\text{cell}) = 0 \). Since \( r, r' \in X_{\text{lin}}^{\text{apx}} \), it follows that the tuple \((r, r', q_1, q_2)\) is an empty corner. Since \( q_1, q_2 \in Y_{\text{lin}}^{\text{opt}} \), a corner constraint has been introduced binding \( q_1 \) and \( q_2 \) and the existence of \((x_2, y_2)\) in the area of interest of \((r, r', q_1, q_2)\) violates that constraint. This establishes \( r = p'_2 \), that is, \((x_2, y_2)\) lies in \( \text{apxcell}(p'_2, \ell_1) \) and, consequently, \( \delta(\text{apxcell}(p'_2, \ell_1)) = 2 \).
Figure 12: Illustration of the proof of Lemma 4.21 directly after the proof of Claim 4.25. Solid lines are from $X_{\text{lin}}^\text{apx} \cup Y_{\text{lin}}^\text{apx}$, dashed lines are from $X_{\text{lin}}^\text{opt} \cup Y_{\text{lin}}^\text{opt}$. The apx-supercells $\text{apxcell}(p'_1, \ell_1)$ and $\text{apxcell}(p'_2, \ell_1)$ and the opt-supercell $\text{optcell}(p_1, q_1)$ are highlighted. $\delta(\text{optcell}(p_1, \ell)) = 1$ are highlighted by red background and $\delta(\text{optcell}(p_1, \ell)) = 2$ by blue background.

For the second statement of the claim, we essentially repeat the reasoning of Claim 4.23. Assume that for every $\ell \in Y_{\text{lin}}^\text{apx}$ with $q_1 \leq \ell < q_2$ we have $\delta(\text{optcell}(p_1, \ell)) \neq 2$ (the second case is analogous). Consider the tuple $(p'_2, q_1, p_2, \ell_2)$. Observe that its area of interest is contained both in the union of opt-supercells $\text{optcell}(p_1, \ell)$ for $q_1 \leq \ell < q_2$ and in the apx-supercell $\text{apxcell}(p'_2, \ell_2)$. Since $\delta(\text{apxcell}(p'_2, \ell_1)) = 2$, for every cell in the area of interest of $(p'_2, q_1, p_2, \ell_2)$ we have $\delta(\text{cell}) = 0$. Since $p'_2 \in X_{\text{lin}}^\text{apx}$ and $\ell_2 \in Y_{\text{lin}}^\text{apx}$, $(p'_2, q_1, p_2, \ell_2)$ is an empty corner and a corner constraint has been introduced binding $q_1$ and $p_2$. However, the existence of $(x_2, y_2)$ in the area of interest of this empty corner violates this corner constraint. This finishes the proof of the claim.

Claim 4.25. There exists

1. a line $\ell \in Y_{\text{lin}}^\text{opt}$ with $q_2 \leq \ell < \ell_1$ and $\delta(\text{optcell}(p_1, \ell)) = 1$;
2. a line $\ell \in Y_{\text{lin}}^\text{opt}$ with $q_1 \leq \ell < \ell_1$ and $\delta(\text{optcell}(p_1, \ell)) = 2$;
3. a line $\ell \in Y_{\text{lin}}^\text{opt}$ with $\ell_1 \leq \ell < q_1$ and $\delta(\text{optcell}(p_1, \ell)) = 1$;
4. a line $\ell \in Y_{\text{lin}}^\text{opt}$ with $\ell_1 \leq \ell \leq q_1$ and $\delta(\text{optcell}(p_1, \ell)) = 2$.

In particular, the alternation of the situation $\sigma = (p_1, p_2, \ell_1, \ell_2)$ is at least four.

Proof. The first part of the claim (the existence of the lines) implies that the alternation of $\sigma$ is at least three. As an alternation of a situation cannot be an odd integer larger than 1 (cf. Lemma 4.2), we infer that the first part of the claim implies the second one. Thus we are left with proving the first part.

We first invoke Claim 4.23 giving the first and third point. If $\delta(\text{optcell}(p_1, q_1)) = 2$, then we are done. In the other case $\delta(\text{optcell}(p_1, q_1)) = 0$, we invoke Claim 4.24 obtaining the lines promised in the second and fourth point.

See Figure 12 which shows the situation in Claim 4.25. By Claim 4.25 for the situation $\sigma$ an alternation
constraint has been added, a filtering step for correct orders of extremal points has been performed, and a number of alternating lines constraints have been added.

If the alternating constraint the situation \( \sigma \) is violated, then we have our desired contradiction. Otherwise, \((\xi^{x,\text{opt}}(p_1), \xi^{x,\text{opt}}(p_2))\) fits the situation \( \sigma \), that is, \( \hat{S}_\sigma = \hat{S}(\xi^{x,\text{opt}}(p_1), \xi^{x,\text{opt}}(p_2)) \).

Let \( \ell_1, \ell_2, \ldots, \ell_{2r} \) be the elements of \( \hat{L}_\sigma \) in the increasing order. By Claim 4.25, \( \ell^2 \leq q_1 < \ell^{2r} \). Let \( i \in \mathbb{N} \), \( 1 < i < 2r \), be the maximum index with \( \ell^i \leq q_1 \). Observe that \( \xi^{y,\text{opt}}(\ell^i) < y_1, y_2 \) and \( y_1, y_2 < \xi^{y,\text{opt}}(\ell^{i+1}) \) while \( \xi^{x,\text{opt}}(p_1) < x_1, x_2 \) and \( x_1, x_2 < \xi^{x,\text{opt}}(p_2) \).

We assume that \( \delta(\text{optcell}(p_1, \ell^i)) = 2 \). The reasoning for \( \delta(\text{optcell}(p_1, \ell^i)) = 1 \) is analogous, but uses the point \( (x_2, y_2) \) instead of \( (x_1, y_1) \) and the alternating lines constraint concerning \( p_2 \) instead of \( p_1 \), while \( \delta(\text{optcell}(p_1, \ell^i)) \neq 0 \) by the definition of \( \hat{L}_\sigma \).

Recall that \( \hat{S}_\sigma = \hat{S}(\xi^{x,\text{opt}}(p_1), \xi^{x,\text{opt}}(p_2)) \). Let \( j \in [r] \) be the index of the block of \( \text{pts}_\sigma(\xi^{x,\text{opt}}(p_1), \xi^{x,\text{opt}}(p_2)) \) that contains \( (x_1, y_1) \). Since \( (x_1, y_1) \in W_1 \) but \( \delta(\text{optcell}(p_1, \ell^i)) = 2 \), we have \( i \neq j \). Observe that \( \xi^{y,\text{opt}}(q_2) \leq \xi^{y,\text{opt}}(\ell^{i+1}) \). Recalling the remaining inequalities, we have \( \xi^{y,\text{opt}}(\ell^i) \leq \xi^{y,\text{opt}}(q_1) < y_1 < \xi^{y,\text{opt}}(q_2) \leq \xi^{y,\text{opt}}(\ell^{i+1}) \). If \( i > j \) and all monotonicity constraints are satisfied, then the alternating lines constraint for \( p_1 \) and line above the \( j \)-th block is violated. Similarly, if \( i < j \) and all monotonicity constraints are satisfied, then the alternating lines constraint for \( p_1 \) and line below the \( j \)-th block is violated. This is the desired contradiction that finishes the proof of Lemma 4.21.

One remark is in order. A meticulous reader can notice that the proof of Lemma 4.21 does not in its guts use the filtering step based on Branching Steps D and E. That is, they are not needed to obtain completeness (the conclusion of Lemma 4.21). However, this filtering step has been pivotal in ensuring that alternating lines contraints are sufficiently simple in the proof of Lemma 4.10.

### 4.9 Wrap up

We wrap up the proof of Theorem 1.1 As already discussed, the branching steps result in \( 2^{O(k^2 \log k)} \log n \) subcases. In each subcase, we perform polynomial-time computation that reduces some domains in filtering steps, possibly discarding the subcase. If the subcase is not discarded, it produces an auxiliary CSP instance with \( k \) variables and a number of constraints. There are \( O(k) \) monotonicity constraints, \( O(k^2) \) corner constraints, and \( O(k^2) \) alternation constraints, each of constant depth. Finally, a situation \( \sigma = (p_1, p_2, \ell_1, \ell_2) \) of alternation \( a \geq 4 \) results in \( O(a) \) alternating lines constraints that can be represented as a tree of size \( O(a) \) via Lemma 4.10. This tree, if translated directly into a forest CSP instance as discussed in Section 3, yields \( O(a) \) variables and constraints. There are \( O(k) \) choices of the line \( p_1 \) (which determines \( p_2 \)) and, for fixed \( p_1 \) and \( p_2 \), the sum of alternations of all situations \( (p_1, p_2, \cdot, \cdot) \) is \( O(k) \). Hence, adding all alternating lines constraints directly into a forest CSP instance yields \( O(k^2) \) constraints and variables. Hence, we obtain a forest-CSP instance of apparent size \( O(k^2) \). This gives fixed-parameter tractability of the OPTIMAL DISCRETIZATION problem by Theorem 3.2 and a running time bound of \( 2^{O(k^2 \log k)} n^O(1) \) by Lemma 3.4.

**Acknowledgements.** Marcin Pilipczuk would like to thank Brian Lavalette and Blair Sullivan for introducing him to the problem and sharing some initial thoughts during his visit at NCSU in February 2018.

We acknowledge the comfortable and inspiring atmosphere of Dagstuhl Seminar 19271, where the details of Section 3 have been discussed.

**References**

[1] N. Alon, R. Yuster, and U. Zwick. Color-coding. *J. ACM*, 42(4):844–856, 1995.

[2] L. Barto, A. A. Krokhin, and R. Willard. Polymorphisms, and how to use them. In A. A. Krokhin and S. Zivny, editors, *The Constraint Satisfaction Problem: Complexity and Approximability*, volume 7 of *Dagstuhl Follow-Ups*, pages 1–44. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2017.
[3] É. Bonnet, P. Giannopoulos, and M. Lampis. On the parameterized complexity of red-blue points separation. In D. Lokshtanov and N. Nishimura, editors, 12th International Symposium on Parameterized and Exact Computation, IPEC 2017, September 6-8, 2017, Vienna, Austria, volume 89 of LIPIcs, pages 8:1–8:13. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2017.

[4] G. Călinescu, A. Dumitrescu, H. J. Karloff, and P. Wan. Separating points by axis-parallel lines. Int. J. Comput. Geometry Appl., 15(6):575–590, 2005.

[5] C. Carbonnel and M. C. Cooper. Tractability in constraint satisfaction problems: a survey. Constraints, 21(2):115–144, 2016.

[6] M. Cygan, F. V. Fomin, L. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh. Parameterized Algorithms. Springer, 2015.

[7] M. Dom, M. R. Fellows, F. A. Rosamond, and S. Sikdar. The parameterized complexity of stabbing rectangles. Algorithmica, 62(1-2):564–594, 2012.

[8] U. M. Fayyad and K. B. Irani. Multi-interval discretization of continuous-valued attributes for classification learning. In R. Bajcsy, editor, Proceedings of the 13th International Joint Conference on Artificial Intelligence. Chambéry, France, August 28 - September 3, 1993, pages 1022–1029. Morgan Kaufmann, 1993.

[9] V. Froese. Fine-Grained Complexity Analysis of Some Combinatorial Data Science Problems. PhD thesis, Technische Universität Berlin, 2018.

[10] J. Kujala and T. Elomaa. Improved algorithms for univariate discretization of continuous features. In J. N. Kok, J. Koronacki, R. L. de Mántaras, S. Matwin, D. Mladenic, and A. Skowron, editors, Knowledge Discovery in Databases: PKDD 2007, 11th European Conference on Principles and Practice of Knowledge Discovery in Databases, Warsaw, Poland, September 17-21, 2007. Proceedings, volume 4702 of Lecture Notes in Computer Science, pages 188–199. Springer, 2007.