# Covering Points with Rectangular Boundaries

**Madhumita Kundu**
University of Bergen, Norway

**Daniel Lokshtanov**
University of California Santa Barbara, CA, USA

**Soumi Nandi**
The Institute of Mathematical Sciences, Chennai, India

**Saket Saurabh**
The Institute of Mathematical Sciences, Chennai, India
University of Bergen, Norway

**Kushal Singanporia**
The Institute of Mathematical Sciences, Chennai, India

## Abstract

Geometric covering problems typically ask for a small family of *geometric* objects whose union contains all input points. In this paper we study a more rigid variant, *boundary covering*, where every point must lie on the boundary of at least one chosen object. Motivated by the framework of Langerman and Morin [Discret. Comput. Geom., 2005] for boundary covering by hyperspheres, we initiate a systematic study of boundary covering by axis-parallel rectangles in the plane.

We first consider the *discrete* setting, where the rectangles must be chosen from a given family. We define Boundary Covering with Discrete Axis-Parallel Rectangles (BCDAPR) as follows: given a point set $P \subseteq \mathbb{R}^2$, a collection $\mathcal{R}$ of axis-parallel rectangles, and an integer $k$, decide whether $P$ can be covered by the boundaries of at most $k$ rectangles from $\mathcal{R}$. We prove that this discrete boundary-covering problem is W[1]-hard when parameterized by $k$.

This motivates the *continuous* variant, where we are allowed to place rectangles freely. We define Boundary Covering with Continuous Axis-Parallel Rectangles (BCCAPR) as follows: given a point set $P \subseteq \mathbb{R}^2$ and an integer $k$, decide whether $P$ can be covered by the boundaries of at most $k$ axis-parallel rectangles. In contrast to the discrete case, we show that BCCAPR is fixed-parameter tractable parameterized by $k$, with running time $2^{\mathcal{O}(k \log k)} \cdot n^{\mathcal{O}(1)}$, where $n = |P|$. Our results does a fine-grained structural analysis of how $k$ rectangles can interact with the point set. On the hardness side, we show that moving from lines to slightly richer shapes already incurs intractability: we prove NP-completeness for boundary covering by axis-aligned $L$-shapes, and then lift it to NP-completeness of BCCAPR. For the algorithm we reduce BCCAPR to at most $2^{\mathcal{O}(k \log k)}$ instances of Distinct Domain Monotone 2-CSP, each solvable in polynomial time.

**2012 ACM Subject Classification** Theory of computation → Fixed parameter tractability; Theory of computation → Computational geometry; Theory of computation → Problems, reductions and completeness

**Keywords and phrases** Geometric Covering, Axis-parallel Rectangles, W[1] and NP Hardness, Fixed Parameter Tractability, CSP

**Digital Object Identifier** 10.4230/LIPIcs.ESA.2026.153

**Related Version** *Full Version*: `https://arxiv.org/abs/2607.08183`

## 1 Introduction

Geometric covering problems ask for a small collection of geometric objects whose union "covers" a given set of points. Geometric covering and packing problems have been extensively studied from the perspectives of algorithms and complexity. Since many of these geometric


set cover problems are NP-hard, they have also been investigated from the viewpoints of approximation algorithms and parameterized complexity [11, 5, 13, 14, 6]. In this paper we focus on one such geometric set cover problem in the realm of parameterized complexity.

A particularly subtle variant is *boundary covering*, where each input point must lie on the *boundary* of at least one chosen object (rather than anywhere in its interior). A key starting point for our work is a result of Langerman and Morin [11], which is widely regarded as one of the seminal contributions to parameterized computational geometry. In Section 5.2 of their paper, they study the following boundary-covering problem for spheres:

Covering Points with Spheres. *Given a set $S$ of $n$ points in $\mathbb{R}^d$, does there exist a subcollection $\mathcal{H}' \subseteq \mathcal{H}$ of size at most $k$ such that each point of $S$ lies on the surface of at least one hypersphere in $\mathcal{H}'$?*

They show fixed-parameter tractability (parameter $k$, for fixed $d$) by casting the problem as an instance of their framework Dim-Set-Cover. The crucial geometric observation enabling this is a *dimension-drop* property: if one considers the families $R_i$ of all $i$-spheres (points, pairs of points, circles, etc.), then the intersection of an $i$-sphere and a $j$-sphere (assuming neither contains the other) is an $\ell$-sphere for some $\ell < \min\{i, j\}$. Thus, intersections strictly reduce dimension, which is exactly the structural condition that powers their algorithmic approach. In the planar case ($d = 2$), this is a problem about covering points by *circle boundaries*. It is easy to misread this as a tractability statement for covering points by disks, but these problems are fundamentally different: requiring points to lie on the boundary is much more rigid than allowing them to lie anywhere in the interior. In fact, the usual disk-cover variants remain W[1]-hard even under strong restrictions (e.g., unit disks; see, e.g., [13]). In particular, unless FPT = W[1], there is no algorithm running in time $f(k)\, n^{\mathcal{O}(1)}$ that, given $n$ points in the plane and an integer $k$, decides whether they can be covered by (at most) $k$ unit disks.

This distinction motivates us to systematically study boundary-covering questions for other geometric families. Langerman and Morin also point out that their dimension-drop methods do not apply to axis-parallel rectangles, since the intersection of two rectangles may again be a rectangle and therefore need not reduce dimension (see conclusion in [11]). Motivated by this limitation, we investigate the boundary-covering analogue for axis-parallel rectangles, formalized as follows.

Boundary Covering with Discrete Axis-Parallel Rectangles (BCDAPR)

**Input:** A set $P$ of $n$ points, a set $\mathcal{R}$ of $m$ axis-parallel rectangles in the plane $\mathbb{R}^2$, and an integer $k \geq 0$.
**Parameter:** $k$.
**Question:** Does there exist a set of at most $k$ rectangles in $\mathcal{R}$ whose *boundaries* cover all points in $P$?

**Roadmap.** We first study the parameterized complexity of the above *discrete* rectangle variant and show that it is W[1]-hard with respect to the natural parameter $k$. This motivates us to consider the *continuous* setting, where rectangles may be placed freely. For this variant we establish NP-completeness, and then show that, despite this hardness, the problem admits a fixed-parameter algorithm parameterized by $k$.

**Context and closest tractable analogue.** While covering point sets by various geometric objects has been studied extensively, we are not aware of prior work on this boundary-covering variant for axis-parallel rectangles. The closest classical tractable special case is boundary

covering by *axis-parallel lines*: a minimum family of horizontal and vertical lines whose union covers all points can be found in polynomial time via a reduction to Bipartite Vertex Cover (equivalently, by computing a maximum matching and applying Kőnig's theorem) [5, 3]. However, once we impose separate upper bounds on the number of lines parallel to the $x$-axis and to the $y$-axis, the problem becomes NP-complete via a simple reduction from Constrained Bipartite Vertex Cover(CBVC) (given a bipartite graph $G = (A, B, E)$ and integers $k_A, k_B$, the question is whether there exists a vertex cover using at most $k_A$ vertices from $A$ and at most $k_B$ vertices from $B$) [10]. Another closely related setting is where we are given a family of segments in the plane and asked to cover all points by this fixed set of segments. This variant was recently studied by Kowalska and Pilipczuk [8], who obtained a detailed parameterized complexity classification, showing that some versions admit FPT algorithms while others are hard.

## 1.1 Our Results

Our first result establishes that Boundary Covering with Discrete Axis-Parallel Rectangles (BCDAPR) is W[1]-hard when parameterized by $k$ even in 2-dimension.

▶ **Theorem 1.** Boundary Covering with Discrete Axis-Parallel Rectangles *is* W[1]*-hard when parameterized by* $k$*.*

Since the discrete rectangle variant is hard, we turn to the *continuous* setting, where rectangles may be placed freely; in particular, we study the following problem.

Boundary Covering with Continuous Axis-Parallel Rectangles (BCCAPR)

**Input:** A finite set $P = \{p_1, \ldots, p_n\}$ of points in the plane $\mathbb{R}^2$, and a nonnegative integer $k \in \mathbb{N} \cup \{0\}$.
**Parameter:** $k$.
**Question:** Does there exist a family $\mathcal{R} = \{R_1, \ldots, R_{k'}\}$ of axis-parallel rectangles with $k' \leq k$ such that every point of $P$ lies on the boundary of at least one rectangle in $\mathcal{R}$?

We remark that Langerman and Morin [11] studied an analogous *continuous* variant for spheres. Their paper explicitly presents an algorithm for the continuous version.

As noted above, boundary covering by *axis-parallel lines* (when lines may be chosen freely) is solvable in polynomial time, whereas the corresponding *discrete* version (where one must choose from a given set of axis-parallel lines) is NP-complete. Our next contribution shows that, even in the continuous setting, moving beyond lines to slightly richer axis-parallel shapes already leads to intractability. We first prove NP-completeness for boundary covering by axis-aligned L-shapes, Boundary Covering with Continuous Axis-Parallel L-shapes (BCCAPL), where an L-shape is the union of one horizontal and one vertical segment sharing an endpoint. We then use this as a starting point to obtain NP-completeness for BCCAPR.

▶ **Theorem 2.** [1] BCCAPL *is NP-complete.*

▶ **Theorem 3.** BCCAPR *is NP-complete.*

[1] Omitted results can be found in attached full version.

Constrained Bipartite Vertex Cover is our starting point for the reduction to BCCAPL [10]. Theorem 2 provides the core hardness gadget: it captures the essential "turning" behavior that rectangle boundaries must simulate. Building on this gadget, our reduction for Theorem 3 encodes each L-shape choice using a constant number of rectangle-boundary constraints while preserving the parameter $k$.

Having established NP-completeness, we study BCCAPR from the perspective of parameterized complexity with respect to the solution size $k$. Our main algorithmic result shows that, despite NP-hardness, the problem is fixed-parameter tractable parameterized by $k$.

▶ **Theorem 4.** BCCAPR *is fixed-parameter tractable when parameterized by $k$ and admits an algorithm with running time $2^{\mathcal{O}(k \log k)} n^{\mathcal{O}(1)}$, where $n$ is the number of input points.*

This result is obtained by carefully analyzing how the $k$ rectangles in a solution can interact with the point set. We first discretize the plane and compute a set of at most $4k$ axis-parallel lines that together contain all input points. If size of the set exceeds $4k$, we directly return `No`. We then guess which side of which rectangle aligns with which chosen line, and how the points on each such line are covered by the horizontal sides of the rectangles aligned with it. Points that are not covered by these aligned rectangles are called *exceptional points*; we show that their total number is bounded by $2k$, and we guess how these exceptional points are covered by sides of rectangles that do not align with the chosen lines. These guesses allow us to encode the problem as an instance of Distinct Domain Monotone 2-CSP, which is known to be solvable in polynomial time [1]. In summary, we obtain at most $2^{\mathcal{O}(k \log k)}$ instances of Distinct Domain Monotone 2-CSP such that the input instance is a `Yes`-instance if and only if at least one of these Distinct Domain Monotone 2-CSP instances is satisfiable. This also illustrates the power of Distinct Domain Monotone 2-CSP as a tool for designing FPT algorithms in computational geometry [1].

## 1.2 Related work

Many classical geometric set cover problems remain hard when parameterized by the number $k$ of objects: for example, covering points with unit squares is W[1]-hard (as shown by Marx, see also later expositions) [13, 8], and related separator-based techniques give $n^{\mathcal{O}(\sqrt{k})}$-time algorithms for covering with disks/squares rather than FPT running times [15]. Covering points by lines also has a rich literature (the general Point Line Cover problem is NP-hard and has been studied from a parameterized/kernelization perspective) [5, 11, 9].

# 2 Notations and Preliminaries

For $m \in \mathbb{N}$, we denote $\{1, \ldots, m\}$ by $[m]$ and $\{0, \ldots, m\}$ by $[m]_0$. An axis-parallel rectangle $R$ in the plane has the following form $R = \{(x, y) \in \mathbb{R}^2 \mid a \leq x \leq b,\ c \leq y \leq d\}$, for some $a, b, c, d \in \mathbb{R}$. Throughout the paper, all rectangles are assumed to be axis-parallel, so we simply refer to them as rectangles. The boundary of a rectangle $R$, denoted by $\mathsf{bd}(R)$, is defined as $\mathsf{bd}(R) = \{(x, y) \in R \mid x = a \text{ or } x = b \text{ or } y = c \text{ or } y = d\}$, that is, $\mathsf{bd}(R)$ consists of four boundary line segments of $R$.

The four lines, namely, $x = a$, $x = b$, $y = c$ and $y = d$ are termed as the *boundary lines* of $R$. Furthermore, we call the sides of a rectangle $R$ as *left, right, bottom* and *top side* of $R$ and the corresponding lines on which they lie, we call them the *left, right, bottom* and *top boundary line* of $R$ respectively. A point $p \in P$ is said to be *covered* by a rectangle $R$ if $p$ lies on the boundary of $R$.

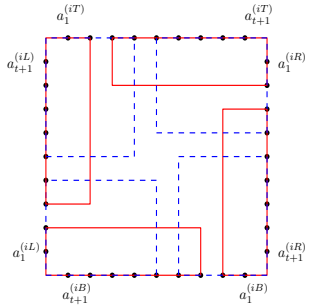



**Figure 1** Variable gadget $SQ_i$.

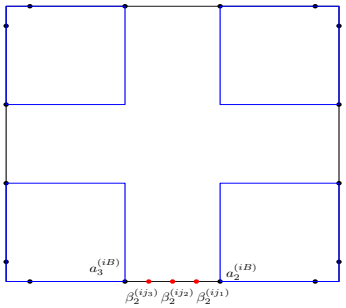



**Figure 2** Constraint points corresponding to domain value $a_2$ in the variable gadget $\mathsf{SQ}_i$.

Let $\pi$ be an ordering of the elements in the set $\{e_1, \ldots, e_n\}$. We write $x \prec_\pi y$ to indicate that element $x$ appears to the left of element $y$ in the order induced by $\pi$. Accordingly, we represent the ordering $\pi$ as $e_1 \prec_\pi e_2 \prec_\pi \cdots \prec_\pi e_n$. When the ordering is clear from context, we omit the subscript and simply write $\prec$.

For the convenience of the reader, the definitions of CSP and Monotone-2-CSP are also included after the references Appendix A and Appendix B.

## 3 Technical overview for W[1]-hardness for BCDAPR

We prove W[1]-hardness of BCDAPR by a parameterized reduction from 3-Regular 2-CSP, where each constraint involves exactly two variables and each variable appears in exactly three constraints. Let $I = (Z, \mathcal{C}, D, \{C_e\}_{e\in\mathcal{C}})$ be a 3-Regular 2-CSP instance, where $Z = \{z_1, z_2, \cdots, z_{k'}\}$ is the set of variables, $D = \{a_1, \ldots, a_t\}$ is the common domain, and $\mathcal{C}$ is a set of $m' = \frac{3k'}{2}$ constraints. We construct a set of points $P$, a family of axis-parallel rectangles $\mathcal{R}$, and a budget $k := 4k' + m' = \frac{11k'}{2}$, such that $I$ is satisfiable if and only if the constructed instance admits a boundary cover by at most $k$ rectangles.

**Variable gadgets.** For each variable $z_i$ we place a square $\mathsf{SQ}_i$ of side length $t + 2$. Along each side of $\mathsf{SQ}_i$ we place $t$ *variable points*, corresponding to the domain values $a_1, \ldots, a_t$ in order, followed by a dummy point. The same ordering is repeated cyclically on all four sides around the boundary of $\mathsf{SQ}_i$. For every $a_j \in D$ we introduce four *variable rectangles* $BL_j^{(i)}, LT_j^{(i)}, TR_j^{(i)}, RB_j^{(i)}$, each uniquely determined by a pair of variable points on two consecutive sides of $\mathsf{SQ}_i$ that form its diagonally opposite corners. These four rectangles are pairwise disjoint and together cover all variable points of $\mathsf{SQ}_i$. Crucially, covering all variable points forces choosing *exactly one* such quadruple: with fewer than four variable rectangles some variable points remain uncovered, and with four rectangles the only way to cover all variable points is to pick one rectangle of each type with a *common index* $j$, i.e., $\{BL_j^{(i)}, LT_j^{(i)}, TR_j^{(i)}, RB_j^{(i)}\}$. Thus, selecting these four rectangles encodes the assignment $z_i = a_j$. See Figure 1.

We arrange the variable gadgets, namely the squares $\mathsf{SQ}_1, \ldots, \mathsf{SQ}_{k'}$, next to each other so that their bottom sides lie on the same horizontal line; see Figure 3.

**Constraint gadgets.** Suppose $z_i$ appears in the three constraints $C_{ij_1}, C_{ij_2}, C_{ij_3}$. For each domain value $a_p \in D$, we place three *constraint points* $\beta_p^{(ij_1)}, \beta_p^{(ij_2)}, \beta_p^{(ij_3)}$ on the bottom side of $\mathsf{SQ}_i$, between the consecutive variable points corresponding to $a_p$ and $a_{p+1}$. By design,


**Figure 3** Arrangement of variable gadgets.

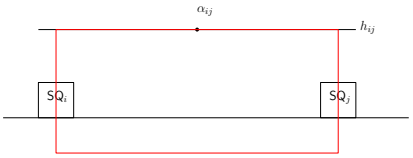



**Figure 4** Constraint guard point $\alpha_{ij}$ corresponding to variables $z_i, z_j$.

when the variable gadget selects the quadruple encoding $z_i = a_p$, these three constraint points are *not* covered by the chosen variable rectangles and must be covered by additional rectangles, see Figure 2 for an illustration.

For each constraint $C_{ij}$ we introduce a dedicated horizontal line segment $h_{ij}$ above the variable gadgets and place on it a *guard point* $\alpha_{ij}$ that cannot be covered vertically (refer to Figure 4). For every satisfying pair $(a_p, a_q)$ of $C_{ij}$, we add a *constraint rectangle* whose boundary covers $\alpha_{ij}$ horizontally and covers $\beta_p^{(ij)}$ and $\beta_q^{(ji)}$ vertically. Because each $\alpha_{ij}$ lies on its own line $h_{ij}$, covering all guard points forces the solution to pick at least one rectangle per constraint. See Figure 5 for illustration.

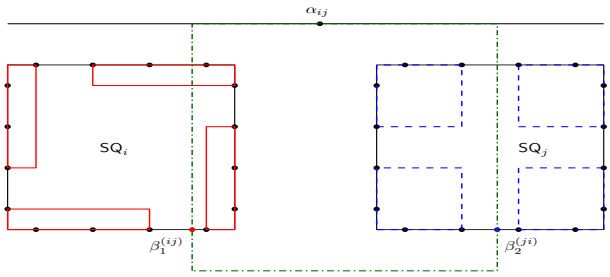



**Figure 5** Constraint rectangle covering constraint points and constraint guard point.

**Budget forcing and correctness.** We set $k = 4k' + m'$. Any feasible solution must cover all variable points, implying at least 4 variable rectangles per gadget and hence at least $4k'$ variable rectangles overall. Likewise, each guard point $\alpha_{ij}$ can only be covered by a constraint rectangle associated with $C_{ij}$, so at least $m'$ constraint rectangles are necessary. Since the total budget is exactly $4k' + m'$, every size-$k$ solution must be *tight*: it selects exactly one quadruple of variable rectangles per variable and exactly one constraint rectangle per constraint. This yields a bijection between solutions and assignments: from a satisfying assignment $\sigma$ we select the corresponding $4k'$ variable rectangles and, for each constraint, the unique rectangle corresponding to the satisfying pair $(\sigma_i, \sigma_j)$; conversely, from any size-$k$ solution we read off a unique value $\pi(i)$ per variable gadget and argue that for every constraint $C_{ij}$ the selected constraint rectangle exists only if $(a_{\pi(i)}, a_{\pi(j)})$ satisfies $C_{ij}$. Hence the constructed BCDAPR instance is a `Yes` instance if and only if the original 3-Regular 2-CSP instance is satisfiable, establishing W[1]-hardness parameterized by $k$.

## 4 An FPT algorithm for BCCAPR

In this section we give an overview of the FPT algorithm for BCCAPR parameterized by $k$.

## 4.1 Discretization of coordinates

We begin by *discretizing the plane* using the coordinate values induced by the input point set $P = \{p_1, \ldots, p_n\} \subseteq \mathbb{R}^2$, where $p_i = (x_i, y_i)$ for each $i \in [n]$. Let $X$ and $Y$ denote the sets of distinct $x$- and $y$-coordinates appearing in $P$, respectively: $X := \{ x_i \mid (x_i, y_i) \in P \}$ and $Y := \{ y_i \mid (x_i, y_i) \in P \}$. Let $\mathsf{Gridpts}_X = \langle \alpha_1, \alpha_2, \ldots, \alpha_{N_x} \rangle$ be the increasing ordering of $X$, and let $\mathsf{Gridpts}_Y = \langle \beta_1, \beta_2, \ldots, \beta_{N_y} \rangle$ be the increasing ordering of $Y$, where $N_x = |X| \leq n$ and $N_y = |Y| \leq n$. Thus $\alpha_1 < \alpha_2 < \cdots < \alpha_{N_x}$ and $\beta_1 < \beta_2 < \cdots < \beta_{N_y}$.

**Discretization.** Given a point set $P \subseteq \mathbb{R}^2$, let $\mathsf{Gridpts}_X = \langle \alpha_1, \ldots, \alpha_{N_x} \rangle$ and $\mathsf{Gridpts}_Y = \langle \beta_1, \ldots, \beta_{N_y} \rangle$ be the increasing orderings of the distinct $x$- and $y$-coordinates appearing in $P$. We draw the vertical lines $x = \alpha_j$ for all $j \in [N_x]$ and the horizontal lines $y = \beta_\ell$ for all $\ell \in [N_y]$. These lines form an axis-parallel grid whose *grid points* are the intersection points $(\alpha_j, \beta_\ell)$, for all $j \in [N_x]$ and $\ell \in [N_y]$. By construction, every point of $P$ lies on a grid point of this grid (indeed, on an intersection of one drawn vertical and one drawn horizontal line). This yields the following lemma.

▶ **Lemma 5** (Grid-aligned solution for rectangles). *Let $(P, k)$ be an instance of* BCCAPR *and let $X = \{x(p) : p \in P\}$ and $Y = \{y(p) : p \in P\}$. Let $\mathsf{Gridpts}_X = \langle \alpha_1 < \alpha_2 < \cdots < \alpha_{N_x} \rangle$ and $\mathsf{Gridpts}_Y = \langle \beta_1 < \beta_2 < \cdots < \beta_{N_y} \rangle$ be the sorted lists of distinct coordinates in $X$ and $Y$. If $(P, k)$ is a* `Yes`*-instance, then there exists a solution $\mathcal{R}$ of size at most $k$ such that for every rectangle $R \in \mathcal{R}$, all four sides of $R$ lie on grid lines of the form $x = \alpha_j$ and $y = \beta_\ell$ (equivalently, all four corners of $R$ are grid points $(\alpha_j, \beta_\ell)$).*

## 4.2 Covering by few lines via Vertex Cover

We now prove that if there is a solution with at most $k$ rectangles, then the points in $P$ can be covered by at most $4k$ grid lines (vertical or horizontal). Consider the grid induced by the coordinate sets $X$ and $Y$ from the discretization step (see Section 4.1). Construct a bipartite graph $G = (V_{\mathsf{vert}} \uplus V_{\mathsf{hor}}, E)$ where

- $V_{\mathsf{vert}}$ has one vertex $v_j$ for each distinct vertical line $x = \lambda_j$, i.e., $V_{\mathsf{vert}} = \{v_j \mid \lambda_j \in X\}$.
- $V_{\mathsf{hor}}$ has one vertex $h_\ell$ for each distinct horizontal line $y = \gamma_\ell$, i.e., $V_{\mathsf{hor}} = \{h_\ell \mid \gamma_\ell \in Y\}$.
- The edge set $E(G)$ consists of edges $e_p = (v_j, h_\ell)$ for each point $p = (\lambda_j, \gamma_\ell) \in P$.

A subset $C \subseteq V_{\mathsf{vert}} \uplus V_{\mathsf{hor}}$ is a *vertex cover* of $G$ if every edge $e_p \in E(G)$ has at least one endpoint in $C$. Now we relate the solution to BCCAPR to a vertex cover of $G$.

▶ **Lemma 6** (Few-line cover of points). *Let $(P, k)$ be an instance of* BCCAPR*. If $(P, k)$ has a solution of size at most $k$ , then the bipartite graph $G$ has a vertex cover $C$ of size at most $4k$. Moreover, such a minimum vertex cover can be found in polynomial time.*

**Solution Supporting Important Lines.** Let $C$ be a vertex cover of $G$ obtained via Lemma 6, with $|C| \leq 4k$. By the construction of $G$, the set $C$ corresponds to at most $4k$ vertical or horizontal lines that together cover all points in $P$. We denote this collection of lines by $\mathsf{ImpLines}$. Clearly $|\mathsf{ImpLines}| \leq 4k$. Since the vertex cover $C$ can be computed algorithmically, the set $\mathsf{ImpLines}$ can be computed as well [7].

## 4.3 Exceptional points: definition and basic bounds

Let $\mathsf{ImpLines}_{\mathsf{hor}}$ and $\mathsf{ImpLines}_{\mathsf{vert}}$ denote the sets of horizontal and vertical lines in $\mathsf{ImpLines}$, respectively. Since $\mathsf{ImpLines} = \mathsf{ImpLines}_{\mathsf{hor}} \uplus \mathsf{ImpLines}_{\mathsf{vert}}$ and $|\mathsf{ImpLines}| \leq 4k$, we have $|\mathsf{ImpLines}_{\mathsf{hor}}| \leq 4k$ and $|\mathsf{ImpLines}_{\mathsf{vert}}| \leq 4k$. We now state a few basic properties of the lines

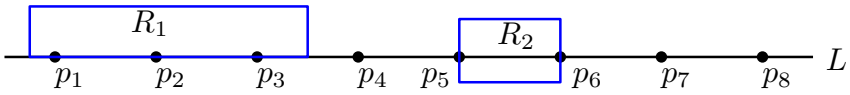



**Figure 6** $p_5, p_6$ (covered by vertical sides of $R_2$) on $L$ are exceptional points w.r.t. $\mathcal{R} = \{R_1, R_2\}$.

in ImpLines. We present them for horizontal lines; the vertical case is completely analogous. Fix a line $L \in \mathsf{ImpLines}_{\mathsf{hor}}$, and let $P_L := P \cap L = \{p_1, \ldots, p_t\}$, where the points are ordered by increasing $x$-coordinate.

▶ **Definition 7** (Exceptional points on a line). *Let $\mathcal{R}$ be a family. A point $p \in P_L$ is* exceptional (w.r.t. $\mathcal{R}$) *if it is covered by some rectangle in $\mathcal{R}$ that has no horizontal side on $L$ (see Figure 6 for reference).*

▶ **Lemma 8** (Few exceptional points per line). *Let $\mathcal{R}$ be a family of at most $k$ rectangles. For any horizontal line $L$, the number of exceptional points of $P_L$ (w.r.t. $\mathcal{R}$) is at most $2k$.*

## 4.4 Skeletons

Recall that by Lemma 6 we can compute, in polynomial time, a set of at most $4k$ grid-aligned lines $\mathsf{ImpLines} = \mathsf{ImpLines}_{\mathsf{vert}} \uplus \mathsf{ImpLines}_{\mathsf{hor}}$ such that every point of $P$ lies on at least one line of ImpLines. Intuitively, a solution by at most $k$ rectangles must "use" many of these lines as rectangle sides; otherwise too many points on a line would have to be covered by rectangles that merely *cross* the line, which is impossible with only $k$ rectangles. We capture this alignment information by the notion of a *skeleton*.

Let $\mathcal{R} = \{R_1, \ldots, R_k\}$ be a family of axis-parallel rectangles, where $R_i = [\ell_i, r_i] \times [b_i, t_i]$, and denote its four sides by $R_i(L)$, $R_i(R)$, $R_i(B)$, and $R_i(T)$ (left, right, bottom, top).

**Supporting line of a side.** Since each side $S$ of an axis-parallel rectangle is a (closed) axis-parallel line segment, it lies on a unique axis-parallel line. We call this line the *supporting line* of $S$ and denote it by $\mathrm{line}(S)$. Formally,

$$\mathrm{line}\big(R_i(L)\big) = \{(x,y) \in \mathbb{R}^2 \mid x = \ell_i\}, \quad \mathrm{line}\big(R_i(R)\big) = \{(x,y) \in \mathbb{R}^2 \mid x = r_i\},$$

$$\mathrm{line}\big(R_i(B)\big) = \{(x,y) \in \mathbb{R}^2 \mid y = b_i\}, \quad \mathrm{line}\big(R_i(T)\big) = \{(x,y) \in \mathbb{R}^2 \mid y = t_i\}.$$

In particular, $\mathrm{line}(S)$ is vertical when $S = R_i(L)$ or $R_i(R)$, and horizontal when $S = R_i(B)$ or $R_i(T)$.

▶ **Definition 9** (Skeleton). *A* skeleton *is a function* $\sigma : \{(i, \mathsf{s}) \mid i \in [k],\ \mathsf{s} \in \{L, R, B, T\}\} \to \mathsf{ImpLines} \cup \{\perp\}$ *such that:*

1. *if $\sigma(i, L) \neq \perp$ then $\sigma(i, L) \in \mathsf{ImpLines}_{\mathsf{vert}}$ (and similarly if $\sigma(i, R) \neq \perp$ then $\sigma(i, R) \in \mathsf{ImpLines}_{\mathsf{vert}}$);*
2. *if $\sigma(i, B) \neq \perp$ then $\sigma(i, B) \in \mathsf{ImpLines}_{\mathsf{hor}}$ (and similarly if $\sigma(i, T) \neq \perp$ then $\sigma(i, T) \in \mathsf{ImpLines}_{\mathsf{hor}}$);*
3. *(No side-identification) $\sigma(i, L) \neq \sigma(i, R)$ whenever both are lines in $\mathsf{ImpLines}_{\mathsf{vert}}$, and $\sigma(i, B) \neq \sigma(i, T)$ whenever both are lines in $\mathsf{ImpLines}_{\mathsf{hor}}$.*
4. *If $\sigma(i, \mathsf{s}) = \perp$, it means that the corresponding side of $R_i$ is* not *aligned with any line of* ImpLines.

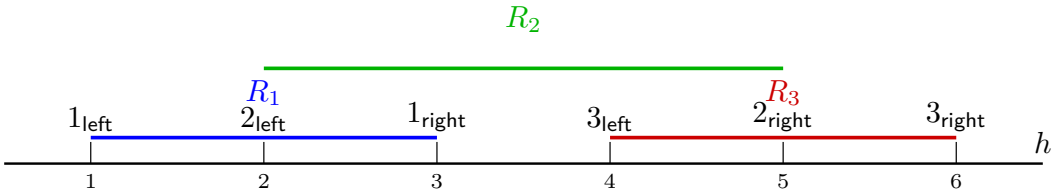



**Figure 7** A horizontal line $h \in \mathsf{ImpLines}_{\mathsf{hor}}$ and three rectangles $R_1, R_2, R_3$ whose top or bottom side lies on $h$ under the fixed skeleton $\sigma$. Each such rectangle induces a segment on $h$ with endpoint symbols $i_{\mathsf{left}}$ and $i_{\mathsf{right}}$, and $\pi(\sigma, h)$ records the left-to-right order of all endpoint symbols, subject to $i_{\mathsf{left}}$ appearing before $i_{\mathsf{right}}$ for every $i \in \mathsf{Ind}(\sigma, h)$. For example, here $\pi(\sigma,h)(1_{\mathsf{left}}) = 1 < \pi(\sigma,h)(2_{\mathsf{left}}) = 2 < \pi(\sigma,h)(1_{\mathsf{right}}) = 3$ and $\pi(\sigma,h)(3_{\mathsf{left}}) = 4 < \pi(\sigma,h)(2_{\mathsf{right}}) = 5 < \pi(\sigma,h)(3_{\mathsf{right}}) = 6$.

## 4.5 Fixing a skeleton and ordering endpoints

From now on, we fix a valid skeleton $\sigma$ as in Definition 9. Please refer to Figure 7 for an illustration. We continue to use the notation $\sigma(i, L)$, $\sigma(i, R)$, $\sigma(i, B)$, and $\sigma(i, T)$ for the values of the skeleton on the four sides of rectangle $R_i$ (left, right, bottom, top). In particular, $\sigma(i, L), \sigma(i, R) \in \mathsf{ImpLines}_{\mathsf{vert}} \cup \{\perp\}$ and $\sigma(i, B), \sigma(i, T) \in \mathsf{ImpLines}_{\mathsf{hor}} \cup \{\perp\}$.

Fix an arbitrary horizontal line $h \in \mathsf{ImpLines}_{\mathsf{hor}}$. The skeleton tells us exactly which rectangles have their *bottom* or *top* side aligned with $h$. We collect their indices as

$$\mathsf{Ind}(\sigma, h) \;:=\; \{\, i \in [k] \mid \sigma(i, B) = h \text{ or } \sigma(i, T) = h \,\}, \qquad m(\sigma, h) \;:=\; |\mathsf{Ind}(\sigma, h)|.$$

For every $i \in \mathsf{Ind}(\sigma, h)$, the intersection of $h$ with the corresponding side of $R_i$ is a horizontal segment and hence has a *left endpoint* and a *right endpoint*. We represent these endpoints symbolically by introducing two endpoint labels per such $i$:

$$\begin{aligned} \mathsf{Endpts}_{\mathsf{left}}(\sigma, h) &:= \{\, i_{\mathsf{left}} \mid i \in \mathsf{Ind}(\sigma, h) \,\}, \\ \mathsf{Endpts}_{\mathsf{right}}(\sigma, h) &:= \{\, i_{\mathsf{right}} \mid i \in \mathsf{Ind}(\sigma, h) \,\}, \\ \mathsf{Endpts}(\sigma, h) &:= \mathsf{Endpts}_{\mathsf{left}}(\sigma, h) \uplus \mathsf{Endpts}_{\mathsf{right}}(\sigma, h). \end{aligned}$$

Clearly, $|\mathsf{Endpts}_{\mathsf{left}}(\sigma, h)| = |\mathsf{Endpts}_{\mathsf{right}}(\sigma, h)| = m(\sigma, h)$ and $|\mathsf{Endpts}(\sigma, h)| = 2m(\sigma, h)$. (We treat the symbols $i_{\mathsf{left}}$ and $i_{\mathsf{right}}$ as abstract labels; later we will guess their geometric order along $h$.)

▶ **Definition 10** (Endpoint ordering on a line). *An* endpoint ordering *on $h$ (with respect to $\sigma$) is a bijection $\pi(\sigma, h) : \mathsf{Endpts}(\sigma, h) \to [\,2m(\sigma, h)\,]$, which induces the total order*

$$\pi(\sigma,h)^{-1}(1) \;\prec\; \pi(\sigma,h)^{-1}(2) \;\prec\; \cdots \;\prec\; \pi(\sigma,h)^{-1}\big(2m(\sigma,h)\big) \tag{1}$$

*on the symbols in $\mathsf{Endpts}(\sigma, h)$.*

▶ **Definition 11** (Valid endpoint ordering). *An endpoint ordering $\pi(\sigma, h)$ is* valid *if for every $i \in \mathsf{Ind}(\sigma, h)$ we have $\pi(\sigma,h)(i_{\mathsf{left}}) < \pi(\sigma,h)(i_{\mathsf{right}})$, that is, the left-endpoint symbol of $i$ appears before its right-endpoint symbol in the order* (1).

## 4.6 Blocks in an endpoint ordering

From now on we fix a skeleton $\sigma$, and for every line in $\mathsf{ImpLines}$ we guess an endpoint ordering. Fix an arbitrary horizontal line $h \in \mathsf{ImpLines}_{\mathsf{hor}}$ and a guess for a *valid* endpoint ordering of $\mathsf{Endpts}(\sigma, h)$, induced by the bijection $\pi(\sigma, h)$. Recall that $m(\sigma, h) = |\mathsf{Ind}(\sigma, h)|$ and $|\mathsf{Endpts}(\sigma, h)| = 2m(\sigma, h)$. For readability, we write $m := m(\sigma, h)$ and $\pi := \pi(\sigma, h)$.

**The Dyck word of an ordering (balanced-parentheses condition).** The ordering $\pi$ induces a word $W = W(\sigma, h) = w_1 w_2 \cdots w_{2m} \in \{\mathsf{left}, \mathsf{right}\}^{2m}$ by setting $w_j = \mathsf{left}$ if $\pi^{-1}(j) \in \mathsf{Endpts}_{\mathsf{left}}(\sigma, h)$ and $w_j = \mathsf{right}$ if $\pi^{-1}(j) \in \mathsf{Endpts}_{\mathsf{right}}(\sigma, h)$. Since $\pi$ is a valid endpoint ordering, every prefix of $W$ contains at least as many left's as right's. Moreover, $W$ contains exactly $m$ occurrences of each symbol. Thus $W$ is a Dyck word (see [2] for a quick definition) over $\Sigma = \{\mathsf{left}, \mathsf{right}\}$.

▶ **Definition 12** (Complete prefixes). *For every $j \in [2m]$, let $L(j) := \#_{\mathit{left}}\big(W[1..j]\big)$ and $R(j) := \#_{\mathit{right}}\big(W[1..j]\big)$. We call the prefix $W[1..j]$* complete *if $L(j) = R(j)$.*

By validity, for all $j \in [2m]$ we have $L(j) \geq R(j)$, and the full word is complete, i.e, $L(2m) = R(2m) = m$. Equivalently, $W[1..j]$ is complete if and only if there is no $i \in \mathsf{Ind}(\sigma, h)$ with $\pi(i_{\mathsf{left}}) \leq j < \pi(i_{\mathsf{right}})$.

▶ **Definition 13** (Minimal complete blocks). *Let $0 = j_0 < j_1 < \cdots < j_q = 2m$ be the indices of the complete prefixes of $W$, defined by letting $j_r$ be the smallest index larger than $j_{r-1}$ with $L(j_r) = R(j_r)$. For each $r \in [q]$, define $\mathsf{block}_r := W[j_{r-1} + 1 \,..\, j_r]$. We call $\mathsf{block}_1, \ldots, \mathsf{block}_q$ the* minimal complete blocks *(equivalently,* minimal Dyck blocks*) of the ordering.*

▶ **Lemma 14** (Unique decomposition into minimal Dyck blocks). *Let $W \in \{\mathit{left}, \mathit{right}\}^*$ be a Dyck word, and let $\mathsf{block}_1, \ldots, \mathsf{block}_q$ be the blocks from Definition 13. Then:*

1. *each $\mathsf{block}_r$ is a nonempty Dyck word and no proper prefix of $\mathsf{block}_r$ is a Dyck word (i.e., $\mathsf{block}_r$ is* minimal*);*
2. *$W = \mathsf{block}_1 \mathsf{block}_2 \cdots \mathsf{block}_q$;*
3. *this decomposition is unique among all decompositions of $W$ into minimal Dyck words.*

**Blocks and endpoint symbols.** Each block $\mathsf{block}_r$ corresponds to a contiguous sub-order of endpoint symbols: it starts at position $j_{r-1} + 1$ and ends at position $j_r$ in the total order induced by $\pi$. Equivalently, the symbols in block $\mathsf{block}_r$ are exactly

$$\pi^{-1}(j_{r-1} + 1) \prec \pi^{-1}(j_{r-1} + 2) \prec \cdots \prec \pi^{-1}(j_r).$$

In particular, $\pi^{-1}(1)$ is the first symbol of $\mathsf{block}_1$, and $\pi^{-1}(2m)$ is the last symbol of $\mathsf{block}_q$.

## 4.7 Gaps in an endpoint ordering

Recall that we have fixed a skeleton $\sigma$ and, for each line in ImpLines, we guess a valid endpoint ordering. Fix a horizontal line $h \in \mathsf{ImpLines}_{\mathsf{hor}}$, and abbreviate $m := m(\sigma, h)$ and $\pi := \pi(\sigma, h)$. Let $W = W(\sigma, h) \in \{\mathsf{left}, \mathsf{right}\}^{2m}$ be the Dyck word induced by $\pi$ as in the previous subsection, and let $W = X_1 X_2 \cdots X_q$ be its (unique) decomposition into minimal Dyck blocks.

**Gaps.** We define the *gaps* of this decomposition to be the $q+1$ regions: one before $X_1$, one between each consecutive pair $X_r, X_{r+1}$, and one after $X_q$. So, we set $\mathsf{GapNum}(\pi) := q + 1$.

**Guessing the number of exceptional points per gap.** Let $K$ be an integer with $0 \leq K \leq 2k$ (intuitively, $K$ is the total number of exceptional points on $h$). A *gap assignment* is a function

$$\mathsf{GapFn} : [\mathsf{GapNum}(\pi)] \to \{0, 1, \ldots, K\} \quad \text{such that} \quad \sum_{j=1}^{\mathsf{GapNum}(\pi)} \mathsf{GapFn}(j) = K. \tag{2}$$

▶ **Definition 15** (Gap vector). *A* gap vector *(for the fixed $(\sigma, h, \pi)$) is a pair $(K, \mathsf{GapFn})$ where $K \in \mathbb{Z}_{\geq 0}$ and $\mathsf{GapFn}$ satisfies* (2)*.*

## 4.8 Exceptional patterns

Fix a skeleton $\sigma$ and a horizontal line $h \in \mathsf{ImpLines}_{\mathsf{hor}}$. Fix a valid endpoint ordering $\pi = \pi(\sigma, h)$ of $\mathsf{Endpts}(\sigma, h)$ and let $W = W(\sigma, h)$ be the induced Dyck word, with minimal Dyck-block decomposition $W = X_1 \cdots X_q$ (cf. Lemma 14). Let $\mathsf{GapNum}(\pi) := q + 1$ be the number of gaps.

**Exceptional points and the gap vector.** Recall that a point of $P_h := P \cap h$ is *exceptional* (w.r.t. $h$) if it is covered by a rectangle whose boundary *does not* have a horizontal side on $h$ (equivalently, the covering occurs via a vertical side crossing $h$). We fix a gap vector $(K, \mathsf{GapFn})$, where $K$ is our guess for the number of exceptional points on $h$, and $\mathsf{GapFn} : [\mathsf{GapNum}(\pi)] \to \{0, 1, \ldots, K\}$ satisfies $\sum_{r=1}^{\mathsf{GapNum}(\pi)} \mathsf{GapFn}(r) = K$ (cf. Definition 15).

**Rectangles that can create exceptional points on $h$.** Let

$$\mathsf{Ind}(\sigma, h) \;:=\; \{\, i \in [k] \mid \sigma(i, B) = h \text{ or } \sigma(i, T) = h \,\}.$$

Rectangles indexed by $\mathsf{Ind}(\sigma, h)$ have a horizontal side on $h$ and thus cover *non-exceptional* points on $h$ horizontally. Only rectangles with no horizontal side on $h$ can create exceptional points. Define $\bar{\mathsf{Ind}}(\sigma, h) := [k] \setminus \mathsf{Ind}(\sigma, h)$. Among these, some rectangles may cross $h$ (and then their vertical sides intersect $h$). We guess the subset $\mathsf{Ind}_{\mathsf{exp}}(\sigma, h) \subseteq \bar{\mathsf{Ind}}(\sigma, h)$ consisting of those indices $i$ for which $R_i$ intersects $h$ (equivalently, $b_i \le h \le t_i$ in coordinates).

▶ **Definition 16** (Exceptional pattern on $h$). *Fix a gap vector $(K, \mathsf{GapFn})$ and a guess $\mathsf{Ind}_{\mathsf{exp}}(\sigma, h) \subseteq [k] \setminus \mathsf{Ind}(\sigma, h)$. An* exceptional pattern *is a function*

$$\mathsf{ExpPat}(\sigma, h) : [K] \;\to\; \mathsf{Ind}_{\mathsf{exp}}(\sigma, h) \times \{L, R\}.$$

*We interpret $\mathsf{ExpPat}(\sigma, h)(r) = (i, L)$ (respectively, $(i, R)$) as the declaration that the $r$-th exceptional point on $h$ from left to right is covered by the left (respectively, right)* vertical *side of rectangle $R_i$.*

## 4.9 Reduction to 2-CSP

We now give the full parameterized reduction from BCCAPR to Monotone 2-CSP. Given an instance $(P, k)$ of BCCAPR, we construct (after a bounded amount of guessing that depends only on $k$) an equivalent Monotone 2-CSP instance.

**Setup.** We assume that we have already carried out the discretization (Lemma 5) and computed, via Lemma 6, a set of grid-aligned lines, namely, $\mathsf{ImpLines} = \mathsf{ImpLines}_{\mathsf{vert}} \uplus \mathsf{ImpLines}_{\mathsf{hor}}$, with $|\mathsf{ImpLines}| \le 4k$, that covers all points of $P$. Moreover, we work only with grid-aligned rectangles, so every rectangle is of the form $[\ell, r] \times [b, t]$ with $\ell, r \in \mathsf{Gridpts}_X$ and $b, t \in \mathsf{Gridpts}_Y$. See Figure 8.

## Global guessing pattern

The reduction proceeds by enumerating the following (all within $k^{\mathcal{O}(k)}$ choices overall):

- **(G1)** a skeleton $\sigma$ as in Definition 9;
- **(G2)** for every line $\lambda \in \mathsf{ImpLines}$, an endpoint ordering $\pi(\sigma, \lambda)$ as in Definition 10;
- **(G3)** for every line $\lambda \in \mathsf{ImpLines}$, a gap vector $(K(\sigma, \lambda), \mathsf{GapFn}(\sigma, \lambda))$ as in Definition 15;
- **(G4)** for every line $\lambda \in \mathsf{ImpLines}$, a set $\mathsf{Ind}_{\mathsf{exp}}(\sigma, \lambda) \subseteq [k] \setminus \mathsf{Ind}(\sigma, \lambda)$ and an exceptional pattern $\mathsf{ExpPat}(\sigma, \lambda)$ (without any extra validity condition; the CSP will enforce feasibility).

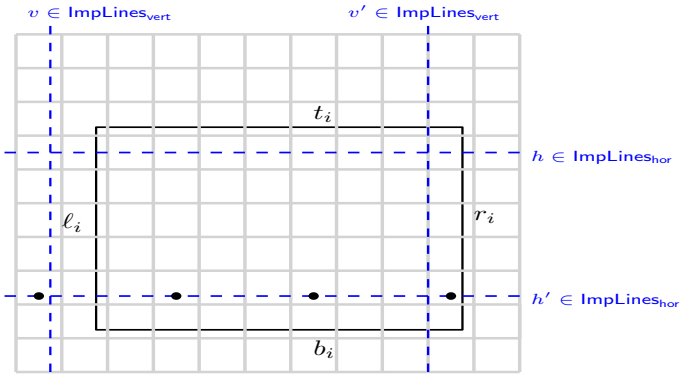



**Figure 8** Example picture: grid-aligned lines in ImpLines (blue), one rectangle $R_i = [\ell_i, r_i] \times [b_i, t_i]$, and points on $h'$. Global view: after discretization, all points lie on a small set of grid-aligned lines ImpLines, and each rectangle is described by four coordinate variables.

For each line $\lambda \in \mathsf{ImpLines}$, let $\pi_\lambda := \pi(\sigma, \lambda)$ be the guessed endpoint ordering, let $\big(K_\lambda, \mathsf{GapFn}_\lambda\big) := \big(K(\sigma,\lambda), \mathsf{GapFn}(\sigma,\lambda)\big)$ be the guessed gap data, and let $\big(\mathsf{Ind}_{\mathsf{exp},\lambda}, \mathsf{ExpPat}_\lambda\big) := \big(\mathsf{Ind}_{\mathsf{exp}}(\sigma,\lambda), \mathsf{ExpPat}(\sigma,\lambda)\big)$ be the guessed exceptional data. We group these per-line guesses into the global objects

$$\Pi := \big(\pi_\lambda\big)_{\lambda \in \mathsf{ImpLines}}, \qquad \Gamma := \big(K_\lambda, \mathsf{GapFn}_\lambda\big)_{\lambda \in \mathsf{ImpLines}}, \qquad \mathcal{E} := \big(\mathsf{Ind}_{\mathsf{exp},\lambda}, \mathsf{ExpPat}_\lambda\big)_{\lambda \in \mathsf{ImpLines}}.$$

We may assume w.l.o.g. that each guessed gap datum in (G3) is *internally consistent*, i.e., for every $\lambda \in \mathsf{ImpLines}$ we have $K_\lambda = \sum_g \mathsf{GapFn}_\lambda(g)$. (If not, we discard the guess immediately.) For each fixed global guess $(\sigma, \Pi, \Gamma, \mathcal{E})$ we build an Monotone 2-CSP instance $\mathcal{I}(\sigma, \Pi, \Gamma, \mathcal{E})$, and accept iff at least one such instance is satisfiable.

## CSP variables (global rectangle coordinates)

We introduce four Monotone 2-CSP variables $\ell_i, r_i, b_i, t_i$ for each rectangle $R_i$. Their domains are subsets of the discretized grids: $D(\ell_i), D(r_i) \subseteq \mathsf{Gridpts}_X$, and $D(b_i), D(t_i) \subseteq \mathsf{Gridpts}_Y$. An assignment to these variables specifies the rectangle $R_i = [\ell_i, r_i] \times [b_i, t_i]$.

**Skeleton as domains.** We enforce the skeleton by restricting domains as follows. Let

$$X_{\mathsf{ImpLines}} := \{\, x(v) \mid v \in \mathsf{ImpLines}_{\mathsf{vert}} \,\} \qquad \text{and} \qquad Y_{\mathsf{ImpLines}} := \{\, y(h) \mid h \in \mathsf{ImpLines}_{\mathsf{hor}} \,\}.$$

Define $X_\perp := \mathsf{Gridpts}_X \setminus X_{\mathsf{ImpLines}}$ and $Y_\perp := \mathsf{Gridpts}_Y \setminus Y_{\mathsf{ImpLines}}$. For each $i \in [k]$, set

$$D(\ell_i) = \begin{cases} \{x(\sigma(i,L))\} & \text{if } \sigma(i,L) \in \mathsf{ImpLines}_{\mathsf{vert}}, \\ X_\perp & \text{if } \sigma(i,L) = \perp, \end{cases}$$

$$D(r_i) = \begin{cases} \{x(\sigma(i,R))\} & \text{if } \sigma(i,R) \in \mathsf{ImpLines}_{\mathsf{vert}}, \\ X_\perp & \text{if } \sigma(i,R) = \perp, \end{cases}$$

$$D(b_i) = \begin{cases} \{y(\sigma(i,B))\} & \text{if } \sigma(i,B) \in \mathsf{ImpLines}_{\mathsf{hor}}, \\ Y_\perp & \text{if } \sigma(i,B) = \perp, \end{cases}$$

$$D(t_i) = \begin{cases} \{y(\sigma(i,T))\} & \text{if } \sigma(i,T) \in \mathsf{ImpLines}_{\mathsf{hor}}, \\ Y_\perp & \text{if } \sigma(i,T) = \perp. \end{cases}$$

Here $x(v)$ is the $x$-coordinate of a vertical line $v$ and $y(h)$ is the $y$-coordinate of a horizontal line $h$.

## Proper rectangles

We define *successor maps* on the ordered grids $\mathsf{Gridpts}_X = \langle \alpha_1 < \cdots < \alpha_{N_x} \rangle$ and $\mathsf{Gridpts}_Y = \langle \beta_1 < \cdots < \beta_{N_y} \rangle$ as follows: $\mathsf{succ}_X$ maps each $x$-grid value to the next grid value to its right, and $\mathsf{succ}_Y$ maps each $y$-grid value to the next grid value above it. Formally,

$$\mathsf{succ}_X(\alpha_j) := \alpha_{j+1} \quad (j \in [N_x - 1]), \qquad \mathsf{succ}_Y(\beta_\ell) := \beta_{\ell+1} \quad (\ell \in [N_y - 1]).$$

We enforce the strict inequalities $\ell_i < r_i$ and $b_i < t_i$ using $\mathsf{succ}_X$ and $\mathsf{succ}_Y$. To ensure these functions are always defined, we restricted $D(\ell_i) \subseteq \{\alpha_1, \ldots, \alpha_{N_x - 1}\}$ and $D(b_i) \subseteq \{\beta_1, \ldots, \beta_{N_y - 1}\}$. Then, for each $i \in [k]$, we add the monotone constraints: $r_i \geq \mathsf{succ}_X(\ell_i)$, $t_i \geq \mathsf{succ}_Y(b_i)$. Since all coordinates take values from the grids, these constraints force $\ell_i < r_i$ and $b_i < t_i$.

## Per-line monotone counting primitives

We will repeatedly need to express, using monotone functions, statements of the form: "how many points of $P$ on a given line lie before a certain coordinate?" For a horizontal line we count along the $x$-axis, and for a vertical line we count along the $y$-axis.

**Horizontal line $h \in \mathsf{ImpLines}_{\mathsf{hor}}$.** Let $P_h := P \cap h$ and $t_h := |P_h|$. For $x \in \mathsf{Gridpts}_X$, define

$$A_h(x) := \big|\{\, p \in P_h : x(p) < x \,\}\big|, \qquad B_h(x) := \big|\{\, p \in P_h : x(p) \leq x \,\}\big|.$$

**Vertical line $v \in \mathsf{ImpLines}_{\mathsf{vert}}$.** Let $P_v := P \cap v$ and $t_v := |P_v|$. For $y \in \mathsf{Gridpts}_Y$, define

$$A_v(y) := \big|\{\, p \in P_v : y(p) < y \,\}\big|, \qquad B_v(y) := \big|\{\, p \in P_v : y(p) \leq y \,\}\big|.$$

All these functions are monotone nondecreasing on their respective finite domains.

## Constraints for one horizontal line $h \in \mathsf{ImpLines}_{\mathsf{hor}}$

Fix a horizontal line $h \in \mathsf{ImpLines}_{\mathsf{hor}}$. Recall

$$\mathsf{Ind}(\sigma, h) = \{\, i \in [k] \mid \sigma(i, B) = h \text{ or } \sigma(i, T) = h \,\}, \qquad m(\sigma, h) = |\mathsf{Ind}(\sigma, h)|.$$

We also recall the endpoint-symbol set $\mathsf{Endpts}(\sigma, h) = \{\, i_{\mathsf{left}}, i_{\mathsf{right}} : i \in \mathsf{Ind}(\sigma, h) \,\}$ and the guessed endpoint ordering $\pi = \pi(\sigma, h) : \mathsf{Endpts}(\sigma, h) \to [\, 2m(\sigma, h) \,]$.

**Endpoint symbols as aliases for rectangle variables.** Endpoint symbols are *combinatorial labels* used only to describe the guessed left-to-right order $\pi(\sigma, h)$. To talk about their $x$-coordinates inside the CSP, we reuse the existing rectangle-boundary variables via the alias map

$$\mathsf{coord}_h : \mathsf{Endpts}(\sigma, h) \to \{\ell_1, r_1, \ldots, \ell_k, r_k\}, \quad \mathsf{coord}_h(i_{\mathsf{left}}) := \ell_i, \quad \mathsf{coord}_h(i_{\mathsf{right}}) := r_i.$$

For example, if $\pi^{-1}(j) = 2_{\mathsf{left}}$ and $\pi^{-1}(j+1) = 5_{\mathsf{right}}$, then the order constraint $\mathsf{coord}_h(\pi^{-1}(j)) \leq \mathsf{coord}_h(\pi^{-1}(j+1))$ is simply $\ell_2 \leq r_5$. See Figure 9.

**(H1) Order constraints induced by $\pi(\sigma, h)$.** For each $j \in [\, 2m(\sigma, h) - 1 \,]$, add

$$\mathsf{coord}_h\big(\pi^{-1}(j)\big) \;\leq\; \mathsf{coord}_h\big(\pi^{-1}(j+1)\big).$$

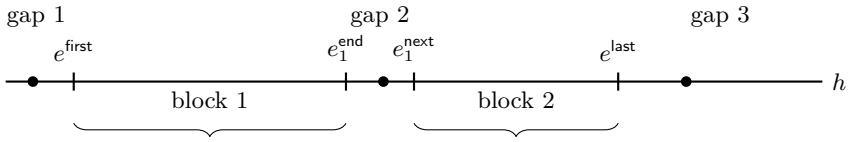



**Figure 9** Endpoint ordering $\pi(\sigma, h)$, minimal Dyck blocks, and the induced gaps on a horizontal line: Schematic: endpoints along $h$ (from $\pi$) decompose into minimal blocks.
Gaps are the regions between blocks; $\mathsf{GapFn}$ prescribes how many points of $P_h$ lie in each gap.

**(H2) Gap constraints via minimal Dyck blocks.** Fix the ordering $\pi$ on $\mathsf{Endpts}(\sigma, h)$. Let $0 = j_0 < j_1 < \cdots < j_q = 2m(\sigma, h)$ be the complete-prefix indices that define the $q$ minimal Dyck blocks (Definition 13 and Lemma 14). These block boundaries carve $h$ into exactly $q + 1$ *gaps*: before the first endpoint; between the end of block $r$ and the next endpoint (for each $r \in [q-1]$); and after the last endpoint. Our guess $\mathsf{GapFn}(\sigma, h) : [q+1] \to \{0, 1, \ldots\}$ specifies how many points of $P_h$ lie in each gap.

**Formal constraints.** Let $e^{\mathsf{first}} := \pi^{-1}(1)$ and $e^{\mathsf{last}} := \pi^{-1}\big(2m(\sigma, h)\big)$.
For each $r \in [q-1]$, define $e_r^{\mathsf{end}} := \pi^{-1}(j_r)$ and $e_r^{\mathsf{next}} := \pi^{-1}(j_r + 1)$.
Write $\mathsf{GapFn} := \mathsf{GapFn}(\sigma, h)$. We add:

$$\begin{aligned} A_h\big(\mathsf{coord}_h(e^{\mathsf{first}})\big) &= \mathsf{GapFn}(1), \\ A_h\big(\mathsf{coord}_h(e_r^{\mathsf{next}})\big) &= B_h\big(\mathsf{coord}_h(e_r^{\mathsf{end}})\big) + \mathsf{GapFn}(r+1) \qquad \text{for all } r \in [q-1], \\ B_h\big(\mathsf{coord}_h(e^{\mathsf{last}})\big) &= |P_h| - \mathsf{GapFn}(q+1). \end{aligned}$$

**(H3) Exceptional points and the exceptional pattern.** We now encode the *exceptional* points on the horizontal line $h$ and how they are covered. Please refer to Figure 10 for an illustration. Recall that $\mathsf{Ind}(\sigma, h)$ is the set of rectangles whose *top* or *bottom* side lies on $h$. Every point of $P_h := P \cap h$ that is not covered by such a horizontal side must instead be covered by a *vertical* side (left/right) of some rectangle whose sides are not aligned with $h$. Accordingly, in the global guessing step we guess: (i) a set $\mathsf{Ind}_{\mathsf{exp}}(\sigma, h) \subseteq [k] \setminus \mathsf{Ind}(\sigma, h)$ of rectangles that may cover exceptional points on $h$, and (ii) an *exceptional pattern* $\mathsf{ExpPat}(\sigma, h)$ specifying, from left to right, which rectangle and which vertical side covers each exceptional point.

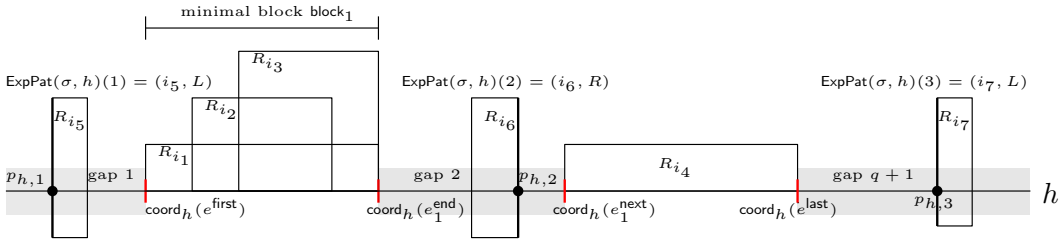
**Idea.** Let $K := K(\sigma, h)$ be the guessed number of exceptional points on $h$. We introduce variables $p_{h,1}, \ldots, p_{h,K}$ that pick $K$ *distinct* points of $P_h$ in left-to-right order. Then we: (a) force $p_{h,1}, \ldots, p_{h,K}$ to fall into the $q + 1$ gaps induced by (H2), in quantities prescribed by $\mathsf{GapFn}(\sigma, h)$; and (b) enforce the guessed pattern $\mathsf{ExpPat}(\sigma, h)$ by making each chosen point coincide with the declared vertical side ($\ell_i$ or $r_i$) and ensuring this side intersects $h$.

**Formal constraints.** Let $\mathsf{Ind}_{\mathsf{exp}}(\sigma, h) \subseteq [k] \setminus \mathsf{Ind}(\sigma, h)$ and $\mathsf{ExpPat}(\sigma, h)$ be the guessed set and pattern on $h$.

*(H3a) Choosing $K$ distinct points of $P_h$ in left-to-right order.* Introduce variables $p_{h,1}, \ldots, p_{h,K}$ with domain $D_h := D(p_{h,r}) := \{\, x(p) \mid p \in P_h \,\} \subseteq \mathbb{N}$ (for all $r \in [K]$). Let 0 and $M_h$ be fixed constants such that $0 < \min D_h$ and $M_h > \max D_h$ (e.g. $M_h := \max(\mathsf{Gridpts}_X) + 1$). Define monotone predecessor/successor maps induced by $D_h$:

$$\mathrm{pred}_h(x) := \max\big(\{d \in D_h : d < x\} \cup \{0\}\big), \qquad \succ_h(x) := \min\big(\{d \in D_h : d > x\} \cup \{M_h\}\big).$$

Enforce distinctness and left-to-right order by adding, for all $r \in [K-1]$, $p_{h,r+1} \geq \succ_h(p_{h,r})$. Thus $p_{h,1}, \ldots, p_{h,K}$ represent $K$ distinct points of $P_h$ in increasing $x$-order.




Each exceptional point lies on $h$ and is forced onto a declared vertical side: $p_{h,u} = \ell_i$ or $p_{h,u} = r_i$, additionally $b_i \le y(h) \le t_i$ to ensure the side crosses $h$

**Figure 10** Exceptional points on $h$: each selected point is placed inside the appropriate gap, and then forced to coincide with the declared vertical side crossing $h$ (thick side), as specified by $\mathsf{ExpPat}(\sigma, h)$.

*(H3b) Placing the $K$ points into the $q+1$ gaps.* Let $q$ be the number of minimal complete blocks of $\pi(\sigma, h)$ as in (H2), and write $\mathsf{GapFn} := \mathsf{GapFn}(\sigma, h) : [q+1] \to \{0, 1, \dots\}$. Define prefix sums $s_g := \sum_{u=1}^{g} \mathsf{GapFn}(u)$ $(g \in [q+1])$, $s_0 := 0$, so that gap $g$ corresponds to indices $r \in [s_{g-1}+1 \,..\, s_g]$. (Here $s_{q+1} = K$ by our consistency assumption on the guess.)

Recall the endpoint symbols from (H2): $e^{\mathsf{first}} := \pi^{-1}(1)$, $e^{\mathsf{last}} := \pi^{-1}\big(2m(\sigma, h)\big)$, $e_r^{\mathsf{end}} := \pi^{-1}(j_r)$, $e_r^{\mathsf{next}} := \pi^{-1}(j_r + 1)$ $(\forall\, r \in [q-1])$. We use the alias map $\mathsf{coord}_h$ to interpret endpoint symbols by rectangle variables.

*Gap 1 (before the first endpoint).* Since $s_1$ is the highest index in the first gap, i.e, in Gap 1, add $p_{h,s_1} \le \mathrm{pred}_h\big(\mathsf{coord}_h(e^{\mathsf{first}})\big)$.

*Intermediate gaps.* For each $r \in [q-1]$, in gap $r+1$, $s_r+1$ is the lowest index and $s_{r+1}$ is the highest index. Add $\succ_h \big(\mathsf{coord}_h(e_r^{\mathsf{end}})\big) \le p_{h,s_r+1}$ and $p_{h,s_{r+1}} \le \mathrm{pred}_h\big(\mathsf{coord}_h(e_r^{\mathsf{next}})\big)$.

*Gap $q+1$ (after the last endpoint).* Since $s_q+1$ is the lowest index in the last gap, i.e, in Gap $q+1$, add $p_{h,s_q+1} \ge \succ_h \big(\mathsf{coord}_h(e^{\mathsf{last}})\big)$.

*(H3c) Enforcing the guessed exceptional pattern.* Finally, enforce that each selected point is covered by the declared vertical side of the declared rectangle. For each $u \in [K]$:

- if $\mathsf{ExpPat}(\sigma, h)(u) = (i, L)$, add $p_{h,u} = \ell_i$ and $b_i \le y(h) \le t_i$;
- if $\mathsf{ExpPat}(\sigma, h)(u) = (i, R)$, add $p_{h,u} = r_i$ and $b_i \le y(h) \le t_i$.

The inequality $b_i \le y(h) \le t_i$ guarantees that the chosen vertical side of $R_i$ intersects the horizontal line $h$. All constraints above are expressible in Monotone 2-CSP since $y(h)$ is a constant and $\mathrm{pred}_h, \succ_h$ are monotone.

## References

1 Akanksha Agrawal, Kristine V. K. Knudsen, Daniel Lokshtanov, Saket Saurabh, and Meirav Zehavi. The parameterized complexity of guarding almost convex polygons. *Discret. Comput. Geom.*, 71(2):358–398, 2024. doi:10.1007/S00454-023-00569-Y.

2 John Baez. Dyck words, 2015. URL: https://blogs.ams.org/visualinsight/2015/07/15/dyck-words/.

3 Daya Ram Gaur and Binay Bhattacharya. Covering points by axis parallel lines. In *Proc. 23rd European Workshop on Computational Geometry*, pages 42–45, 2007.

4 Venkatesan Guruswami, Bingkai Lin, Xuandi Ren, Yican Sun, and Kewen Wu. Parameterized inapproximability hypothesis under exponential time hypothesis. In Bojan Mohar, Igor Shinkar, and Ryan O'Donnell, editors, *Proceedings of the 56th Annual ACM Symposium on Theory of Computing, STOC 2024, Vancouver, BC, Canada, June 24-28, 2024*, pages 24–35. ACM, 2024. doi:10.1145/3618260.3649771.

5 Refael Hassin and Nimrod Megiddo. Approximation algorithms for hitting objects with straight lines. *Discret. Appl. Math.*, 30(1):29–42, 1991. doi:10.1016/0166-218X(91)90011-K.

6 Jan Kára and Jan Kratochvíl. Fixed parameter tractability of independent set in segment intersection graphs. In Hans L. Bodlaender and Michael A. Langston, editors, *Parameterized and Exact Computation, Second International Workshop, IWPEC 2006, Zürich, Switzerland, September 13-15, 2006, Proceedings*, volume 4169 of *Lecture Notes in Computer Science*, pages 166–174. Springer, 2006. doi:10.1007/11847250_15.

7 Bernhard Korte and Jens Vygen. *Combinatorial optimization: theory and algorithms*. Springer, 2008.

8 Katarzyna Anna Kowalska and Michal Pilipczuk. Parameterized and approximation algorithms for coverings points with segments in the plane. In Olaf Beyersdorff, Mamadou Moustapha Kanté, Orna Kupferman, and Daniel Lokshtanov, editors, *41st International Symposium on Theoretical Aspects of Computer Science, STACS 2024, Clermont-Ferrand, France, March 12-14, 2024*, volume 289 of *LIPIcs*, pages 47:1–47:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPIcs.STACS.2024.47.

9 Stefan Kratsch, Geevarghese Philip, and Saurabh Ray. Point line cover: The easy kernel is essentially tight. *ACM Trans. Algorithms*, 12(3):40:1–40:16, 2016. doi:10.1145/2832912.

10 Sy-Yen Kuo and W Kent Fuchs. Efficient spare allocation for reconfigurable arrays. *IEEE Design & Test of Computers*, 4(1):24–31, 2007.

11 Stefan Langerman and Pat Morin. Covering things with things. *Discret. Comput. Geom.*, 33(4):717–729, 2005. doi:10.1007/S00454-004-1108-4.

12 Daniel Lokshtanov, M. S. Ramanujan, Saket Saurabh, and Meirav Zehavi. Parameterized complexity and approximability of directed odd cycle transversal. In Shuchi Chawla, editor, *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*, pages 2181–2200. SIAM, 2020. doi:10.1137/1.9781611975994.134.

13 Dániel Marx. Efficient approximation schemes for geometric problems? In Gerth Stølting Brodal and Stefano Leonardi, editors, *Algorithms - ESA 2005, 13th Annual European Symposium, Palma de Mallorca, Spain, October 3-6, 2005, Proceedings*, volume 3669 of *Lecture Notes in Computer Science*, pages 448–459. Springer, 2005. doi:10.1007/11561071_41.

14 Dániel Marx. Parameterized complexity of independence and domination on geometric graphs. In Hans L. Bodlaender and Michael A. Langston, editors, *Parameterized and Exact Computation, Second International Workshop, IWPEC 2006, Zürich, Switzerland, September 13-15, 2006, Proceedings*, volume 4169 of *Lecture Notes in Computer Science*, pages 154–165. Springer, 2006. doi:10.1007/11847250_14.

15 Dániel Marx and Michal Pilipczuk. Optimal parameterized algorithms for planar facility location problems using voronoi diagrams. *ACM Trans. Algorithms*, 18(2):13:1–13:64, 2022. doi:10.1145/3483425.


## A Definition of 3-Regular 2-CSP

We now define the source problem for our W[1]-hardness. Toward that we first define constraint satisfaction problems (CSPs) of arity two (also called binary constraints).) We follow the notation and definitions of the seminal paper of Guruswami et al. [4]. Formally, a CSP instance $G$ is a quadruple $(V(G), E(G), D, \{C_e\}_{e \in E(G)})$, where:

- $V(G)$ is the set of variables.
- $E(G)$ is the set of constraints. Each constraint $e = \{u_e, v_e\} \in E(G)$ has arity 2 and is related to two distinct variables $u_e, v_e \in V(G)$. The *constraint graph* is the undirected graph on the vertices $V(G)$ and the edges $E(G)$. Note that we allow multiple constraints between the same pair of variables and thus the constraint graph may have parallel edges.
- $D$ is for the alphabet of each variable in $V(G)$. We use $D = [n]$.
- Given a constraint $e \in E(G)$, $C_e \subseteq D \times D = [n] \times [n]$. Furthermore, given $\{C_e\}_{e \in E(G)}$, we can define $\{\Pi_e\}_{e \in E(G)}$, the set of validity functions of constraints. Given a constraint $e \in E(G)$, the validity function $\Pi_e(\cdot, \cdot) \colon D \times D \to \{0, 1\}$ checks whether the constraint $e$ between $u_e$ and $v_e$ is satisfied. That is, $\Pi_e(\cdot, \cdot)$ assigns 1 if and only of the tuple is in $C_e$.

We use $|G| = (|V(G)| + |E(G)|) \cdot |D|$ to denote the *size* of a CSP instance $G$.

**Assignment and Satisfaction Value.** An *assignment* is a function $\sigma \colon V(G) \to D$ that assigns to each variable a value from the alphabet. The *satisfaction value* for a mapping $\sigma$, denoted as $\mathsf{val}(G, \sigma)$, represents the proportion of constraints that $\sigma$ satisfies, i.e.,

$$\mathsf{val}(G, \sigma) = \frac{1}{|E|} \sum_{e \in E} \Pi_e(\sigma(u_e), \sigma(v_e)).$$

The satisfaction value for $G$, denoted by $\mathsf{val}(G)$, is the highest satisfaction value across all mappings, i.e., $\mathsf{val}(G) = \max_{\sigma \colon V(G) \to \Sigma} \mathsf{val}(G, \sigma)$. We define an assignment $\sigma$ as a *solution* to a CSP instance $G$ if $\mathsf{val}(G, \sigma) = 1$, and say $G$ is *satisfiable* if and only if $G$ has a solution. When the context is clear, $\sigma$ is omitted in the constraint description; therefore, $\Pi_e(u_e, v_e)$ represents $\Pi(\sigma(u_e), \sigma(v_e))$.

We will be working with the 3-Regular 2-CSP problem. An input to this problem consists of a 2CSP $G$ with $k$ variables over size-$n$ alphabets, where the constraint graph $G$ is 3-regular. The question is whether $G$ is satisfiable.

▶ **Proposition 17** ([4, 12]). *3-Regular 2-CSP is* W[1]*-hard when parameterized by the number of variables.*

In short, we consider 3-Regular 2-CSP, where every variable appears in exactly three constraints and every constraint involves exactly two variables.

## B Distinct Domain Monotone 2-CSP

The input to this problem is a set $Z = \{z_1, z_2, \cdots, z_{n'}\}$ of variables, a (finite) domain $D_i \subseteq \mathbb{N}$ for the variable $z_i$, for each $i \in [n']$, and a set $C$ of $m'$ constraints, where each constraint is of the following form: $c = [z_i \diamond f(z_j)]$, where $i, j \in [n']$, $f : D_j \to \mathbb{N}$ is a monotone function, and $\diamond \in \{\leq, \geq, =\}$. The objective is to check if there is an assignment $\mathsf{asg} : Z \to \mathbb{N}$ such that: i) for each $i \in [n']$, $\mathsf{asg}(z_i) \in D_i$, and ii) each constraint in $C$ is satisfied, i.e., for each $c = [z_i \diamond f(z_j)] \in C$, $\mathsf{asg}(z_i) \diamond f(\mathsf{asg}(z_j))$ is true.

In the above problem definition, for simplicity in its usage, we allow domains to be empty sets, in which case we trivially have a no-instance of the problem. The problem Monotone 2-CSP is a special case of Distinct Domain Monotone 2-CSP, where the domains of all the variables are the same. A polynomial time algorithm for Monotone 2-CSP can be obtained via a simple reduction to 2-SAT [1]. A very minor modification to this algorithm for Monotone 2-CSP results in a polynomial time algorithm for Distinct Domain Monotone 2-CSP, which is stated in the following proposition.

▶ **Proposition 18** ([1]). Distinct Domain Monotone 2-CSP *has a polynomial time algorithm.*