Parameterized Approximation for Maximum Weight Independent Set of Rectangles and Segments

Jana Cslovjecsek
EPFL, Lausanne, Switzerland

Michał Pilipczuk
Institute of Informatics, University of Warsaw, Poland

Karol Węgrzycki
Saarland University and Max Planck Institute for Informatics, Saarbrücken, Germany

Abstract

In the Maximum Weight Independent Set of Rectangles problem (MWISR) we are given a weighted set of n axis-parallel rectangles in the plane. The task is to find a subset of pairwise non-overlapping rectangles with the maximum possible total weight. This problem is NP-hard and the best-known polynomial-time approximation algorithm, due to Chalermsook and Walczak [SODA 2021], achieves approximation factor $\mathcal{O}(\log \log n)$. While in the unweighted setting, constant factor approximation algorithms are known, due to Mitchell [FOCS 2021] and to Gálvez et al. [SODA 2022], it remains open to extend these techniques to the weighted setting.

In this paper, we consider MWISR through the lens of parameterized approximation. Grandoni, Kratsch and Wiese [ESA 2019] gave a $(1-\varepsilon)$-approximation algorithm running in $k^{\mathcal{O}(k/\varepsilon^8)}n^{\mathcal{O}(1/\varepsilon^8)}$ time, where k is the number of rectangles in an optimum solution. Unfortunately, their algorithm works only in the unweighted setting and they left it as an open problem to give a parameterized approximation scheme in the weighted setting.

We give a parameterized approximation algorithm for MWISR that given a parameter $k \in \mathbb{N}$, finds a set of non-overlapping rectangles of weight at least $(1-\varepsilon)\mathrm{opt}_k$ in $2^{\mathcal{O}(k \log(k/\varepsilon))}n^{\mathcal{O}(1/\varepsilon)}$ time, where opt_k is the maximum weight of a solution of cardinality at most k. We also propose a parameterized approximation scheme with running time $2^{\mathcal{O}(k^2 \log(k/\varepsilon))}n^{\mathcal{O}(1)}$ that finds a solution with cardinality at most k and total weight at least $(1-\varepsilon)\mathrm{opt}_k$ for the special case of axis-parallel segments.

2012 ACM Subject Classification Theory of computation → Approximation algorithms analysis; Theory of computation → Parameterized complexity and exact algorithms

Keywords and phrases parameterized approximation, Maximum Weight Independent Set, rectangles, segments

Digital Object Identifier 10.4230/LIPIcs.ESA.2024.43

Related Version *Full Version*: https://arxiv.org/pdf/2212.01620 [10]

Funding This work is a part of projects BOBR (Michał Pilipczuk) and TIPEA (Karol Węgrzycki) that have received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreements no. 948057 and 850979, respectively).

1 Introduction

In the field of parameterized complexity the goal is to design an algorithm that is efficient not only in terms of the input size, but also in terms of auxiliary *parameters*. On the other end of the spectrum, in the field of approximation algorithms the goal is to design an algorithm that returns a solution that is only slightly worse than the optimum one. These

two notions are traditional frameworks to deal with NP-hard problems. Recently, researchers started to combine the two concepts and try to design approximation algorithms that run in parameterized time. Ideally, given $\varepsilon > 0$ and a parameter $k \in \mathbb{N}$, for example the size of the desired solution, one seeks an algorithm with running time of the form $f(k,\varepsilon)n^{g(\varepsilon)}$ for some functions $f(k,\varepsilon)$ and $g(\varepsilon)$, which returns a $(1+\varepsilon)$-approximate solution. Such an algorithm is called *parameterized approximation scheme* (PAS).

In this paper, we continue this line of work and apply it to a fundamental geometric packing problem. In the MAXIMUM WEIGHT INDEPENDENT SET OF RECTANGLES (MWISR) problem we are given a set $\mathcal{D}$ consisting of n axis-parallel rectangles in the plane alongside with a weight function $\omega\colon \mathcal{D} \to \mathbb{R}$. Each rectangle $R \in \mathcal{D}$ is a closed set of points $[x_1, x_2] \times [y_1, y_2]$ fully characterized by the positions of its four corners. A feasible solution $\mathcal{S} \subseteq \mathcal{D}$ to the MWISR problem consists of rectangles that are pairwise disjoint, i.e., for any two different $R, R' \in \mathcal{S}$ we have $R \cap R' = \emptyset$; we also call such a solution an *independent set*. The objective is to find a feasible solution of maximum total weight. In this paper, we consider a parameterized setting of the problem. We use parameter $k \in \mathbb{N}$ to denote the *cardinality* of the solution. Then $\mathrm{opt}_k(\mathcal{D})$ denotes the maximum possible weight of an independent set in $\mathcal{D}$ whose cardinality is at most k.

MWISR is a fundamental problem in geometric optimization. It naturally arises in various applications, such as map labeling [2, 12], data mining [16], routing [21], or unsplittable flow routing [5]. MWISR is well-known to be NP-hard [14], and it admits a QPTAS [1]. The currently best approximation factor achievable in polynomial time is $\mathcal{O}(\log\log(n))$ [8]. From the parameterized perspective, it is known that the problem is W[1]-hard when parameterized by k, the number of rectangles in the solution, even in the unweighted setting and when all the rectangles are squares [22]. Therefore, it is unlikely that there is an exact algorithm with a running time of the form $f(k)n^{\mathcal{O}(1)}$, even in this restricted setting. In particular, this also excludes any $(1-\varepsilon)$-approximation algorithm running in $f(\varepsilon)n^{\mathcal{O}(1)}$ time [4, 7]. We note that in the case of weighted squares, there is a PTAS with running time of the form $n^{g(\varepsilon)}$ [13].

Approximating MWISR becomes much easier in the unweighted setting. With this restriction, even constant-factor approximation algorithms for MWISR are known [24, 17], and there is a QPTAS (with a better running time [9] than in unweighted setting). Very recently, these algorithms have been generalized: an $8d/3$-approximation in $(nd)^{\mathcal{O}(d4^d)}$-time for the maximum independent set of convex polygons with at most d-direction has been considered [18] (see also [11] for $\mathrm{OPT}^{\varepsilon}$-approximation for arbitrary polygons).

Grandoni, Kratsch and Wiese [19] were the first to consider parameterized approximation for the MWISR problem. They gave a parameterized approximation scheme for unweighted MWISR running in $k^{\mathcal{O}(k/\varepsilon^8)}n^{\mathcal{O}(1/\varepsilon^8)}$ time. As an open problem, they asked if one can also design a PAS in the weighted setting.

▶ **Open Question 1** ([19]). *Does* MAXIMUM INDEPENDENT SET OF RECTANGLES *admit a parameterized approximation scheme in the weighted setting?*

Our contribution. In this paper we provide the following result:

▶ **Theorem 1.1.** *Suppose $\mathcal{D}$ is a set of axis-parallel rectangles in the plane with positive weights. Then given k and $\varepsilon > 0$, one can in $2^{\mathcal{O}(k\log(k/\varepsilon))}|\mathcal{D}|^{\mathcal{O}(1/\varepsilon)}$ time find an independent set in $\mathcal{D}$ of weight at least $(1-\varepsilon)\mathrm{opt}_k(\mathcal{D})$.*

Note that there is a caveat in the formulation above: the returned solution may actually have cardinality larger than k, but there is a guarantee that it will be an independent set. Ideally, we would like the algorithm to return a solution of weight at least $(1-\varepsilon)\mathrm{opt}_k(\mathcal{D})$

and of cardinality at most k. At this point, we are able to provide such an algorithm only in the restricted case of axis-parallel segments (see Theorem 1.2 below), but let us postpone this discussion till later and focus now on Theorem 1.1. Observe here that the issue with solutions of cardinality larger than k becomes immaterial in the unweighted case, hence Theorem 1.1 applied to the unweighted setting solves the problem considered by Grandoni et al. [19] and actually improves the running time of their algorithm.

We now briefly describe the technical ideas behind Theorem 1.1. Similar to Grandoni et al. [19], the starting point is a polynomial-time construction of a grid such that each rectangle in $\mathcal{D}$ contains at least one gridpoint. However, in order to take care of the weights, our grid is of dimensions $(2k^2/\varepsilon) \times (2k^2/\varepsilon)$. Moreover, already in this step, we may return an independent set of weight at least $(1-\varepsilon)\mathrm{opt}_k$ that consists of more than k rectangles. This is the only step where the algorithm may return more than k rectangles.

After this step, the similarities to the algorithm of Grandoni et al. [19] end. We introduce the notion of the *combinatorial type* of a solution. This is simply a mapping from each rectangle in the solution to the set of all gridpoints contained in it (see Definition 3.3). Observe that since the size of the grid is bounded by a function of k and ε, we can afford to guess (by branching into all possibilities) the combinatorial type of an optimum solution in $f(k,\varepsilon)$ time, for some function $f(\cdot,\cdot)$. Notice that there may be many different rectangles matching the type of a rectangle from the optimum solution. However, it is possible that such a rectangle overlaps with neighboring rectangles (and violates independence). Therefore, we need constraints that prevent rectangles from overlapping. For this, we construct an instance of Arity-2 Valued Constraint Satisfaction Problem (2-VCSP) based on the guessed combinatorial type.

Next, we observe that this instance induces a graph that is "almost" planar, hence we may apply a variant of Baker's shifting technique [3]. This technique (which we discuss in detail in Section 3) allows us to divide the instance into many independent instances of 2-VCSP while removing only $\varepsilon \cdot \mathrm{opt}_k$ weight from the optimum solution. Moreover, each of these independent instances induces a graph of bounded treewidth, and hence can be solved exactly in $|\mathcal{D}|^{\mathcal{O}(1/\varepsilon)}$ time. This concludes a short sketch of our approach.

Let us return to the apparent issue that our algorithm may return a solution of cardinality larger than k. This may happen in the very first step of the procedure, during the construction of the grid. By employing a completely different technique, we can circumvent this problem in the restricted setting of axis-parallel segments and prove the following result.

▶ **Theorem 1.2.** *Suppose $\mathcal{D}$ is a set of axis-parallel segments in the plane with positive weights. Then given k and $\varepsilon > 0$, one can in $2^{\mathcal{O}(k^2 \log(k/\varepsilon))}|\mathcal{D}|^{\mathcal{O}(1)}$ time find an independent set in $\mathcal{D}$ of cardinality at most k and weight at least $(1-\varepsilon)\mathrm{opt}_k(\mathcal{D})$.*

Let us remark that we do not know whether in Theorem 1.2, it is necessary to rely on approximation: to the best of our knowledge, it is open whether finding a maximum weight set of k disjoint axis-parallel segments can be solved exactly in fixed-parameter time, parameterized by k. Kára and Kratochvíl [20] and Marx [23] independently solved the unweighted case: they proved that the problem of finding a maximum cardinality independent set of axis-parallel segments admits an FPT algorithm. However, their approach heavily relies on the assumption of unit weights, as in this setting finding any set of k disjoint segments allows one to immediately conclude the search. Our approach is very different, and in fact we show that finding a maximum weight set of k disjoint axis-parallel segments admits an algorithm with running time $W^{\mathcal{O}(k^2)}|\mathcal{D}|^{\mathcal{O}(1)}$, where W is the number of distinct weights present among the segments.

We proceed with an outline of the proof of Theorem 1.2 and highlight some technical ideas. First, we modify the instance so that the number of different weights is bounded. This is done through guessing the largest weight of a rectangle in an optimum solution and rounding the weights down. This is the only place where we lose accuracy on the optimal solution. In other words, the algorithm is fixed-parameter tractable in k and the number of distinct weights $W = (k/\varepsilon)^{\mathcal{O}(1)}$.

With this assumption, we then construct a grid with $\mathcal{O}(k^2)$ lines hitting every segment of the instance. We say that the grid is *nice* with respect to a segment I, if I contains a grid point; equivalently, I is nice if it is hit by two orthogonal lines of the grid. Observe that the constructed grid is not necessarily nice for every segment of the instance. We adapt the previously introduced notion of the *combinatorial type* in order to also accommodate segments which do not contain a grid point. This is done by mapping the segment to its four neighboring grid lines instead of the grid points contained inside the segment. Further, the weight of the segment is added to its combinatorial type. Similarly to before, the combinatorial type of a segment only depends on the grid size and the number of distinct weights. This allows to guess (by branching into all possibilities) the combinatorial type of the optimum solution $\mathcal{S}$ in $k^{\mathcal{O}(1)} \cdot W$ time.

The goal is to construct a grid which is nice with respect to all segments of an optimal solution $\mathcal{S}$. For this, we start by guessing the combinatorial type of all nice segments of an optimal solution $\mathcal{S}$. Then, we incrementally guess the combinatorial type of the next heaviest segment in $\mathcal{S}$ for which the grid is not yet nice. For each such combinatorial type, we find all possible candidate segments and add at most k lines to the grid G. This ensures that a correct candidate segments are hit in both directions. Repeating this procedure at most k times we end up with a grid which is nice with respect to all the segments of $\mathcal{S}$.

Given such a grid, it remains to guess the combinatorial type of all segments in $\mathcal{S}$ and solve the resulting instance. This can be done either greedily or by observing that the problem can be modeled as a 2-CSP instance whose constraint graph is a union of paths. Both these cases work due to the fact that the segments only interact with each other when they lie on the same grid line.

2 Preliminaries

Through the paper, we silently assume that every weight is positive (because we work on maximization, we can simply disregard objects of non-positive weights). A *tree decomposition* of a graph H is a tree T together with a function bag that maps nodes of T to subsets of vertices of H, called *bags*. The following conditions must be satisfied:

- for every vertex u of H, the nodes of T whose bags contain u must form a connected, nonempty subtree of T; and
- for every edge uv of H, there must exist a node of T whose bag contains both u and v.

The *width* of a tree decomposition (T, bag) is $\max_{x \in V(T)} |\mathsf{bag}(x)| - 1$. The *treewidth* of H is the minimum possible width of a tree decomposition of H. By $\mathrm{dist}_H(u, v)$ we mean the distance between vertices u and v in a graph H.

We use well-known results about Arity-2 Valued Constraint Satisfaction Problems (2-VCSPs). For an intuitive description of 2-VCSP, reader is invited to see [15]. An instance of 2-VCSP is a finite set of *variables* X, a domain $D(x)$ for each variable $x \in X$, and a set C of *constraints*. Each constraint $c \in C$ binds an ordered pair $(x_1^{(c)}, x_2^{(c)}) \in X \times X$ of variables in X (not necessarily distinct) and is given by a mapping $f_c \colon D(x_1^{(c)}) \times D(x_2^{(c)}) \to \mathbb{R}$. We assume that f_c is given as a set of pairs $\{(\mathbf{d}, f_c(\mathbf{d})) \colon \mathbf{d} \in D(x_1^{(c)}) \times D(x_2^{(c)})\}$. The goal

is to compute the maximum value of the function $f(\mathbf{u}) \coloneqq \sum_{c \in C} f_c(\mathbf{u}|_{\mathbf{x}_c})$, over all possible assignments $\mathbf{u} \in \prod_{x \in X} D(x)$ of values in respective domains to variables in X. The value $f(\mathbf{u})$ will be called the *revenue* of the assignment $\mathbf{u}$.

Observe that each instance of 2-VCSP induces an undirected graph, called the *Gaifman graph*: the vertex set is the set of variables X, and for every pair of distinct variables $x, y \in X$, there is an edge xy if and only if there is a constraint $c \in C$ such that $\mathbf{x}_c = (x, y)$. Given a class $\mathcal{H}$ of graphs, we can define a restriction of 2-VCSP to $\mathcal{H}$ by focusing only on instances whose Gaifman graph is in $\mathcal{H}$. In this paper we focus only on instances of 2-VCSP where the Gaifman graph has bounded treewidth. In this setting, it is well-known that a standard dynamic programming solves 2-VCSP efficiently [15][1].

▶ **Theorem 2.1** ([15]). 2-VCSP *can be solved in time* $\Delta^{\mathcal{O}(t)} \cdot |X|^{\mathcal{O}(1)}$ *when the Gaifman graph has treewidth at most* t *and all domains are of size at most* Δ.

In Section 4 we also use standard 2-CSPs. These can be modeled by 2-VCSPs where all the constraints are hard: revenue functions f_c assign only values 0 (the constraint is satisfied) or $-\infty$ (the constraint is not satisfied). The task is to find a variable assignment that satisfies all constraints, that is, yields revenue 0.

Proofs of statements marked with (★) are deferred to the full-version of this paper [10].

3 Axis-parallel rectangles

In this section we prove Theorem 1.1. Therefore, we fix the given set $\mathcal{D}$ of weighted axis-parallel rectangles. For a rectangle $R \in \mathcal{D}$, the weight of R is $\omega(R) \in \mathbb{R}$. By $\mathrm{opt}_k(\mathcal{D})$ we denote the maximum possible weight of a set consisting of at most k disjoint rectangles in $\mathcal{D}$. We also fix any *optimum solution*, that is, a set $\mathcal{S} \subseteq \mathcal{D}$ of cardinality at most k satisfying $\omega(\mathcal{S}) = \mathrm{opt}_k(\mathcal{D})$.

We start with a simple preprocessing on $\mathcal{D}$. First, we guess a rectangle $R_{\max} \in \mathcal{S}$ with maximum weight among all rectangles of $\mathcal{S}$. This can be done with an extra overhead of $\mathcal{O}(n)$ in the running time. Observe that $\omega(R_{\max}) \geqslant \mathrm{opt}_k(\mathcal{D})/k$. Further, we remove from $\mathcal{D}$ every rectangle of weight larger than $\omega(R_{\max})$ and every rectangle of weight not exceeding $\varepsilon\omega(R_{\max})/k$; let the obtained instance be $\mathcal{D}'$. Observe that this operation does not decrease the optimum significantly, as none of the former rectangles and at most k of the latter rectangles could be used in $\mathcal{S}$. More precisely, we have

$$\mathrm{opt}_k(\mathcal{D}') \geqslant \mathrm{opt}_k(\mathcal{D}) - k \cdot \frac{\varepsilon \cdot \omega(R_{\max})}{k} \geqslant (1-\varepsilon)\mathrm{opt}_k(\mathcal{D}).$$

After this preprocessing, the optimum decreased by at most $\varepsilon \cdot \mathrm{opt}_k(\mathcal{D})$. This concludes the description of preprocessing. From now on, we silently assume that our instance is $\mathcal{D} \coloneqq \mathcal{D}'$.

3.1 Constructing a grid

We first introduce relevant terminology.

[1] Freuder [15] actually considered only the unweighted 2-CSP, however, as pointed out in, e.g., [6, 26], this dynamic-programming approach can be adapted to the weighted setting. Also, Freuder assumes that a suitable tree decomposition is given on input. Such a tree decomposition can be provided within the stated time complexity by, for instance, the 4-approximation algorithm of Robertson and Seymour [25].

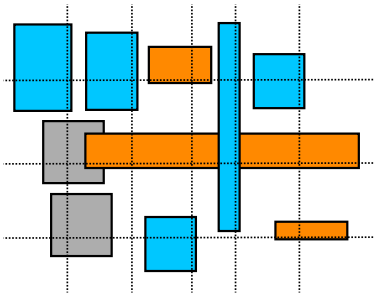

Figure 1 The grid constructed after applying the greedy procedure. Rectangles R_i^{ver} are blue-filled, and rectangles R_i^{hor} are orange-filled. Observe that every rectangle is hit by at least one grid point.

▶ **Definition 3.1.** *A* grid *is a finite set of horizontal and vertical lines in the plane. The* size $|G|$ *of a grid G is the total number of lines it contains. A* grid point *of G is the intersection of a horizontal and a vertical line of G. The set of grid points of G is denoted by* $\mathsf{points}(G)$*. The lines of the grid divide the plane into* grid cells*. Thus, each grid cell is a rectangle, possibly with one or two sides extending to infinity, and at most four* corners*: the grid points lying on its boundary.*

A grid G is good *for a set of axis-parallel rectangles $\mathcal{D}$, if for every rectangle $R \in \mathcal{D}$ there is a grid point of G contained in R.*

As mentioned in Section 1, our search for an optimal solution pivots around a bounded size grid that is good for the optimum solution. The construction of this grid is encapsulated in the following lemma.

▶ **Lemma 3.2.** *Suppose we are given a set $\mathcal{D}$ of weighted axis-parallel rectangles and let $\Delta(\mathcal{D})$ be the ratio between lowest and highest weight in $\mathcal{D}$. Then, supposing $\Delta(\mathcal{D}) \geqslant \varepsilon/k$ for some $\varepsilon > 0$, one can, in polynomial time, either*

- *compute a grid G of size $|G| \leqslant \frac{2k^2}{\varepsilon}$ that is good for $\mathcal{D}$, or*
- *return an independent set $\mathcal{I} \subseteq \mathcal{D}$ with $\omega(\mathcal{I}) \geqslant \mathrm{opt}_k(\mathcal{D})$.*

Proof. We construct the grid G by first constructing the vertical lines of G, and then with basically the same procedure we add the horizontal lines of G. For the construction of the vertical lines, we iteratively pick vertically disjoint rectangles in a greedy fashion. For every rectangle $R \in \mathcal{D}$, select a point $p_R \in R$ very close to the top-right corner of R. We start with $\mathcal{D}_1 \coloneqq \mathcal{D}$. In iteration $i \in \mathbb{N}$, we select a rectangle $R_i^{\text{ver}} \in \mathcal{D}_i$ for which $p_i \coloneqq p_{R_i^{\text{ver}}}$ is the leftmost among rectangles of $\mathcal{D}_i$. (In case of ties, select any of the tying rectangles.) Then, add the vertical line ℓ_i^{ver} which contains p_i to the grid. Next, delete every rectangle from $\mathcal{D}_i$ intersecting ℓ_i^{ver}, thus obtaining the next set $\mathcal{D}_{i+1}$. We repeat this procedure until no more rectangles are left in $\mathcal{D}_i$. To finish the construction of G, repeat the above algorithm in the orthogonal direction, thus selecting vertically disjoint rectangles R_i^{hor} and adding to G horizontal lines ℓ_i^{hor}. This concludes the construction of G; see Figure 1 for an illustration.

Trivially, the above algorithm runs in polynomial time. Moreover, it returns a good grid since every rectangle in $\mathcal{D}$ is intersected by some horizontal and some vertical line from G. So if $|G| \leqslant \frac{2k^2}{\varepsilon}$, we can just return G as the output of the algorithm.

It remains to show that if $|G| > \frac{2k^2}{\varepsilon}$, then we can find an independent set of weight at least $\mathrm{opt}_k(\mathcal{D})$. Assuming that $|G| > \frac{2k^2}{\varepsilon}$, either the vertical or the horizontal run of the greedy algorithm returned more than $\frac{k^2}{\varepsilon}$ lines. Without loss of generality assume that the vertical run gave rectangles $R_1^{\text{ver}}, \ldots, R_m^{\text{ver}}$ for some $m > \frac{k^2}{\varepsilon}$. Note that these rectangles form

an independent set, because after iteration $i \in [m]$ all rectangles with left side to the left of ℓ_i are removed. As we assumed that the ratio between lowest and highest weight of a rectangle in $\mathcal{D}$ is at least ε/k, we may estimate the weight of $\{R_1^{\mathrm{ver}}, \ldots, R_m^{\mathrm{ver}}\}$ as follows:

$$\sum_{i=1}^{m} \omega(R_i^{\mathrm{ver}}) \geqslant m \cdot \frac{\varepsilon \cdot \omega(R_{\max})}{k} \geqslant k \cdot \omega(R_{\max}) \geqslant \mathrm{opt}_k(\mathcal{D}),$$

where $R_{\max}$ is the rectangle of highest weight in $\mathcal{D}$. Therefore, the rectangles $R_1^{\mathrm{ver}}, \ldots, R_m^{\mathrm{ver}}$ form a feasible output for the second point of the lemma statement. ◀

The first step of the algorithm is to run the procedure of Lemma 3.2. If this procedure returns an independent set of weight at least $\mathrm{opt}_k(\mathcal{D})$, we just return it as a valid output and terminate the algorithm. Otherwise, from now on we may assume that we have constructed a grid G of size at most $2k^2/\varepsilon$ and this grid is good for $\mathcal{S}$.

3.2 Combinatorial types

Next, we define the notion of the combinatorial type of a rectangle with respect to a grid. This can be understood as a rough description of the position of the rectangle with respect to the grid.

▶ **Definition 3.3** (Combinatorial Type). *Let G be a grid. For an axis-parallel rectangle R, we define the* combinatorial type $T(R)$ *of R with respect to G as*

$$T_G(R) \coloneqq R \cap \mathsf{points}(G).$$

In other words, $T_G(R)$ is the set of grid points of G contained in R. For a set $\mathcal{S}$ of axis-parallel rectangles, the combinatorial type *of $\mathcal{S}$ is $T_G(\mathcal{S})$, that is, the image of $\mathcal{S}$ under T_G. By Λ_k^G we denote the set of all possible combinatorial types with respect to G of sets $\mathcal{S}$ consisting of at most k axis-parallel rectangles.*

Observe that if a grid is small, there are only few combinatorial types on it.

▶ **Lemma 3.4.** *For every grid G and positive integer k, we have $|\Lambda_k^G| \leqslant 2^{\mathcal{O}(k \log |G|)}$. Moreover, given G and k, Λ_k^G can be constructed in time $2^{\mathcal{O}(k \log |G|)}$.*

Proof. The combinatorial type of any axis-parallel rectangle R can be completely characterized by four lines (or lack thereof) in G: the left-most and the right-most vertical line of G intersecting R, and the top-most and the bottom-most horizontal line of G intersecting R. Hence, there are at most $(|G|+1)^4$ candidates for the combinatorial type of a single rectangle. It follows that the number of combinatorial types of sets of at most k rectangles is bounded by

$$1 + (|G|+1)^4 + (|G|+1)^8 + \ldots + (|G|+1)^{8k} \in 2^{\mathcal{O}(k \log |G|)}.$$

To construct Λ_k^G in time $2^{\mathcal{O}(k \log |G|)}$, just enumerate all possibilities as above. ◀

The next step of the algorithm is as follows. Recall that we work with a grid G of size at most $2k^2/\varepsilon$ that is good for $\mathcal{S}$. By Lemma 3.4, we can compute Λ_k^G in time $2^{\mathcal{O}(k \log(k/\varepsilon))}$ and we have $|\Lambda_k^G| \leqslant 2^{\mathcal{O}(k \log(k/\varepsilon))}$. Hence, by paying a $2^{\mathcal{O}(k \log(k/\varepsilon))}$ overhead in the time complexity, we can guess $\mathcal{T} \coloneqq T_G(\mathrm{opt}_k(G))$, that is, the combinatorial type of the optimum solution. Hence, from now on we assume that the combinatorial type $\mathcal{T}$ is fixed. Since $\mathcal{S}$ is an independent set and G is good for $\mathcal{S}$, we may assume that sets in $\mathcal{T}$ are pairwise disjoint and nonempty.

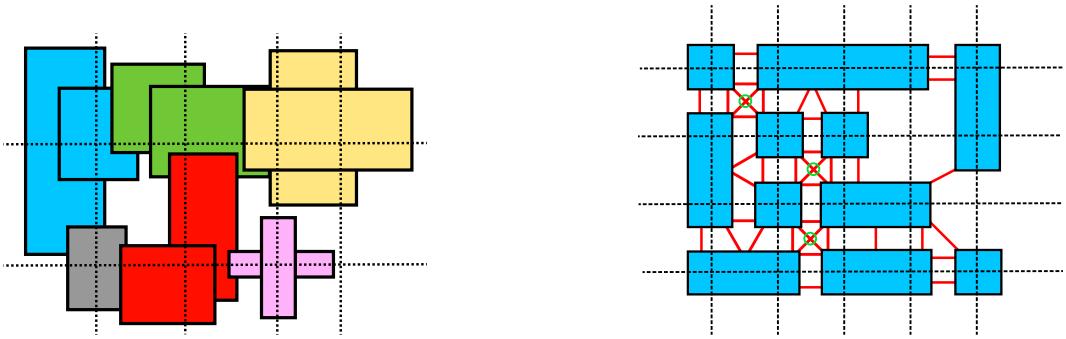

Figure 2 Left Figure: The instance after guessing the combinatorial type $\mathcal{T}$. Rectangles that match the same type $A \in \mathcal{T}$ are filled with the same color. Each variable corresponds to a different rectangle of opt_k, equivalently to a different type $A \in \mathcal{T}$, equivalently to a different color in the figure. The domain of a variable consists of all rectangles in the corresponding color. Right Figure: The Gaifman graph H of the constructed 2-VCSP instance $I_{\mathcal{T}}$. The vertices are depicted in blue and the edges are depicted in thick red. The graph $H^{\bullet}$ is constructed from H by introducing a new vertex at the intersection of every crossing (hence in the Figure we need to add green stroked vertices).

3.3 Reduction to 2-VCSP

We say that a rectangle $R \in \mathcal{D}$ *matches* a subset of grid points $A \subseteq \mathsf{points}(G)$ if $T_G(R) = A$, that is, $R \cap \mathsf{points}(G) = A$. By $\mathcal{D}_A \subseteq \mathcal{D}$ we denote the set of rectangles from $\mathcal{D}$ that match A.

Based on the combinatorial type $\mathcal{T}$ we define an instance $I_{\mathcal{T}}$ of 2-VCSP as follows. The set of variables is $\mathcal{T}$. For every $A \in \mathcal{T}$, the domain of A is $\mathcal{D}_A \cup \{\perp\}$. That is, the set of rectangles from $\mathcal{D}$ that match A plus a special symbol $\perp$ denoting that no rectangle matching A is taken in the solution. Also, for every $A \in \mathcal{T}$ we add a unary[2] constraint c_A on A with associated revenue function $f_{c_A} : \mathcal{D}_A \cup \{\perp\} \to \mathbb{R}$ defined as $f_{c_A}(R) = \omega(R)$ for each $R \in \mathcal{D}_A$ and $f_{c_A}(\perp) = 0$.

It remains to define binary constraints binding pairs of distinct variables in $I_{\mathcal{T}}$. Two distinct grid points of G are *adjacent* if they lie on the same or on consecutive horizontal lines of G, and on the same or on consecutive vertical lines of G. We put a binary constraint $c_{A,B}$ binding variables $A \in \mathcal{T}$ and $B \in \mathcal{T}$ if there exist grid points $p \in A$ and $q \in B$ that are adjacent. The revenue function for $c_{A,B}$ is defined as follows: for $R_A \in \mathcal{D}_A \cup \{\perp\}$ and $R_B \in \mathcal{D}_B \cup \{\perp\}$, we set

$$f_{c_{A,B}}(R_A, R_B) = \begin{cases} -\infty & \text{if } R_A \in \mathcal{D}_A \text{ and } R_B \in \mathcal{D}_B \text{ intersect;} \\ 0 & \text{otherwise.} \end{cases}.$$

In other words, $c_{A,B}$ is a hard constraint: we require that the rectangles assigned to A and B are disjoint (or one of them is nonexistent), as otherwise the revenue is $-\infty$. This concludes the construction of the instance of $I_{\mathcal{T}}$; clearly, it can be done in polynomial time.

The instance $I_{\mathcal{T}}$ is constructed so that it corresponds to the problem of selecting disjoint rectangles from $\mathcal{D}$ that match the combinatorial type $\mathcal{T}$. This is formalized in the following statement.

[2] Formally, in the definition of 2-VCSP we allowed only binary constraints, but unary constraints – constraints involving only one variable – can be modelled by binary constraints binding a variable with itself.

▷ Claim 3.5. If $\mathcal{S} \subseteq \mathcal{D}$ is an independent set of rectangles of combinatorial type $\mathcal{T}$, then there exists a solution to $I_{\mathcal{T}}$ with revenue equal to $\omega(\mathcal{S})$. Conversely, if there exists a solution to $I_{\mathcal{T}}$ with revenue $r \geqslant 0$, then there exists an independent set $\mathcal{S} \subseteq \mathcal{D}$ of weight r and cardinality at most k.

Proof. For the first implication, we construct an assignment $\mathbf{u}\colon \mathcal{T} \to \mathcal{D}$ by setting, for each $A \in \mathcal{T}$, $\mathbf{u}(A)$ to be the unique rectangle $R \in \mathcal{S}$ for which $T_G(R) = A$. To see that the revenue of $\mathbf{u}$ is equal to $\omega(\mathcal{S})$, note that for every $A \in \mathcal{T}$ the unary constraint c_A yields revenue $\omega(\mathbf{u}(A))$, while all binary constraints yield revenue 0, because the rectangles are pairwise disjoint.

For the second implication, let $\mathcal{S} \subseteq \mathcal{D}$ be the image of the assignment $\mathbf{u}$ (possibly with $\perp$ removed). Clearly, $|\mathcal{S}| \leqslant |\mathcal{T}| \leqslant k$. Since $\mathbf{u}$ yields a nonnegative revenue, all binary constraints must give revenue 0, hence $\omega(\mathcal{S})$ is equal to the revenue of $\mathbf{u}$, that is, to r. It remains to argue that $\mathcal{S}$ is an independent set. For this, take any distinct $A, B \in \mathcal{T}$; we need to argue that in case when rectangles $R_A \coloneqq \mathbf{u}(A)$ and $R_B \coloneqq \mathbf{u}(B)$ are both not equal to $\perp$, they are disjoint. Suppose, for contradiction, that R_A and R_B have some common point x. Let Q be the cell of the grid G that contains x. Since $x \in R_A$ and A is nonempty (recall that this is the assumption about all the sets in $\mathcal{T}$, following from G being good for $\mathrm{opt}_k(\mathcal{D})$), A must contain at least one corner of Q, say p. Similarly, B contains a corner of Q, say q. Note that p and q are adjacent grid points, hence in $I_{\mathcal{T}}$ there is a constraint $c_{A,B}$ that yields revenue $-\infty$ in the case when the rectangles assigned to A and B intersect. As this is the case in $\mathbf{u}$, we obtain a contradiction with the assumption $r \geqslant 0$. ◁

3.4 Almost planarity of the Gaifman graph

Let H be the Gaifman graph of $I_{\mathcal{T}}$; see Figure 2 for an example. Recall that the vertex set of H is $\mathcal{T}$, and distinct $A, B \in \mathcal{T}$ are considered adjacent in H iff there is a grid cell Q of G such that both A and B contain a corner of Q. Without loss of generality we assume that H is connected, as otherwise we solve $I_{\mathcal{T}}$ by treating each connected component separately and joining the solutions.

Note that the graph H is not necessarily planar, as there might be crossings within cells; this happens when all four corners belong to different elements of $\mathcal{T}$. However, the intuition is that the crossings within cells are the only problem, hence H is almost planar. We would like to apply Baker's technique on H. We do it in an essentially direct way, except that we need to be careful about the aforementioned crossings. For this, the following construction will be useful.

Call a grid cell Q a *cross* if Q has four corners and all those four corners belong to pairwise different elements of $\mathcal{T}$. Note that then all those four elements form a clique in H. Construct a graph $H^\bullet$ from H as follows: add every cross Q to the vertex set, make it adjacent to all four elements of $\mathcal{T}$ containing the corners of Q, remove the edge connecting the elements of $\mathcal{T}$ containing the top-left and the bottom-right corner of Q, and to the same for the top-right and bottom-left corners.

The reader may imagine $H^\bullet$ as obtained from H by introducing a new vertex at the intersection of diagonals within every cross Q; this new vertex is identified with Q. See Figure 2. So we have the following observation.

▷ Claim 3.6. The graph $H^\bullet$ is planar.

Proof. Let $H_0^\bullet$ be the graph consisting of the grid points of G where two grid points are adjacent if they are consecutive on the same line of G, plus we add a new vertex for every cell of G and make it adjacent to all the corners of this cell. Clearly, $H_0^\bullet$ is planar. Now, $H^\bullet$ can be obtained from $H_0^\bullet$ as follows:

- contract every $A \in \mathcal{T}$ to a single vertex;
- remove every element of $\mathsf{points}(G) - \bigcup \mathcal{T}$; and
- for every grid cell Q of G that is not a cross, either contract the vertex corresponding to Q onto any of its neighbors, or remove it if it has no neighbors.

So $H^\bullet$ is a minor of a planar graph, hence it is planar as well. ◁

We also have the following simple claim, which follows easily from the construction.

▷ Claim 3.7. For all $A, B \in \mathcal{T}$, $\mathrm{dist}_{H^\bullet}(A, B) \leqslant 2 \cdot \mathrm{dist}_H(A, B)$.

Proof. By repeated use of triangle inequality along a shortest path connecting A and B, it suffices to argue the following: if A and B are adjacent in H, then they are at distance at most 2 in $H^\bullet$. For this, observe that either A and B are still adjacent in $H^\bullet$, or they contain two opposite corners of some cross Q, which becomes their common neighbor in $H^\bullet$. ◁

We now apply Baker's technique. Select any $A \in \mathcal{T}$ and partition $\mathcal{T}$ into layers according to the distance in H from A: for a nonnegative integer t, layer $\mathcal{L}_t$ consists of all those vertices $B \in \mathcal{T}$ for which $\mathrm{dist}_H(A, B) = t$. Note that layers $\mathcal{L}_t$ form a partition of $\mathcal{T}$ due to the assumption that H is connected. The following observation is crucial. It follows from applying standard treewidth-radius bounds for planar graphs in subgraphs of $H^\bullet$, and translating the outcome to H.

▶ **Lemma 3.8 (★).** *For all integers $0 \leqslant i < j$, the treewidth of $H[\mathcal{L}_i \cup \mathcal{L}_{i+1} \cup \ldots \cup \mathcal{L}_j]$ is bounded by $\mathcal{O}(j - i)$.*

3.5 Proof of Theorem 1.1

We are ready to finish the proof of Theorem 1.1. The steps performed so far were as follows:

- We guessed a rectangle $R_{\max} \in \mathcal{S}$ (optimum solution) and removed all rectangles of weight larger than $\omega(R_{\max})$ or not exceeding $\varepsilon \cdot \omega(R_{\max})/k$. This induced a loss of at most $\varepsilon \cdot \mathrm{opt}_k(\mathcal{D})$ on the optimum.
- We applied the algorithm of Lemma 3.2. This way, we either find an independent set with a suitably large weight, or we construct a grid G of size $|G| \leqslant 2k^2/\varepsilon$.
- We used Lemma 3.4 to guess, by branching into $2^{\mathcal{O}(k \log(k/\varepsilon))}$ possibilities, the combinatorial type $\mathcal{T}$ of an optimum solution.
- We constructed a 2-VCSP instance $I_{\mathcal{T}}$ corresponding to the type $\mathcal{T}$.

By Claim 3.5, it remains to find a solution to $I_{\mathcal{T}}$ that yields revenue at least $(1 - \varepsilon)\mathrm{opt}(I_{\mathcal{T}})$, where $\mathrm{opt}(I_{\mathcal{T}})$ is the maximum possible revenue in $I_{\mathcal{T}}$. (Note that by retracing previous steps, this results in finding a solution to the original instance of MWISR of weight at least $(1 - 2\varepsilon)\mathrm{opt}_k(\mathcal{D})$, so at the end we need to apply the reasoning to ε scaled by a factor of $1/2$.)

As argued before, we may assume that H, the Gaifman graph of $I_{\mathcal{T}}$, is connected. We partition $\mathcal{T}$ into layers $\{\mathcal{L}_t \colon t = 0, 1, 2, \ldots\}$ as in Section 3.4. Let $\ell := \lceil 1/\varepsilon \rceil$, and define

$$\mathcal{M}_r := \bigcup_{t \equiv r \bmod \ell} \mathcal{L}_t \qquad \text{for all } r \in \{0, 1, \ldots, \ell - 1\}.$$

Note that $\{\mathcal{M}_r \colon r \in \{0, 1, \ldots, \ell - 1\}\}$ is a partition of $\mathcal{T}$.

Let $\mathbf{u}$ be an optimum solution to $I_{\mathcal{T}}$. As it is always possible to assign $\perp$ to every element of $\mathcal{T}$, we have $f(\mathbf{u}) \geqslant 0$, in particular all (hard) binary constraints are satisfied under f. Therefore, $f(\mathbf{u}) = \sum_{r=0}^{\ell-1} f(\mathbf{u}|_{\mathcal{M}_r})$. Since $\ell \geqslant 1/\varepsilon$, there exists $r_0 \in \{0, 1, \ldots, \ell-1\}$ such that $f(\mathbf{u}|_{\mathcal{M}_{r_0}}) \leqslant \varepsilon \cdot f(\mathbf{u})$. The algorithm guesses, by branching into ℓ possibilities, the value of r_0.

Let $I'_{\mathcal{T}}$ be the 2-VCSP instance obtained from $I_{\mathcal{T}}$ by deleting all variables contained in $\mathcal{M}_{r_0}$. Observe that we have $\mathrm{opt}(I'_{\mathcal{T}}) \geqslant (1-\varepsilon) \cdot \mathrm{opt}(I_{\mathcal{T}})$, since $\mathbf{u}$ restricted to the variables of $I'_{\mathcal{T}}$ yields revenue at least $(1-\varepsilon) \cdot \mathrm{opt}(I_{\mathcal{T}})$. Further, every solution to $I'_{\mathcal{T}}$ can be lifted to a solution to $I_{\mathcal{T}}$ of the same revenue by just mapping all variables of $\mathcal{M}_{r_0}$ to $\perp$. Hence, to find an optimum solution to the instance $I'_{\mathcal{T}}$.

For this, observe that the Gaifman graph of $I'_{\mathcal{T}}$ is equal to $H - \mathcal{M}_{r_0}$. This graph is the disjoint union of several subgraphs, each contained in the union of at most $\ell - 1$ consecutive layers $\mathcal{L}_t$. By Lemma 3.8 we infer that the treewidth of $H - \mathcal{M}_{r_0}$ is bounded by $\mathcal{O}(\ell) = \mathcal{O}(1/\varepsilon)$. We apply Theorem 2.1 to solve $I'_{\mathcal{T}}$ optimally in time $|\mathcal{D}|^{\mathcal{O}(1/\varepsilon)} \cdot k^{\mathcal{O}(1)}$. Together with the previous guessing steps, this gives time complexity $2^{\mathcal{O}(k \log(k/\varepsilon))} \cdot |\mathcal{D}|^{\mathcal{O}(1/\varepsilon)}$ in total, as wanted. This concludes the proof of Theorem 1.1.

4 Axis-parallel segments

In this section we prove Theorem 1.2. We use the same notation as in Section 3: $\mathcal{D}$ is the given set of axis-parallel segments, $\omega \colon \mathcal{D} \to \mathbb{R}$ is the weight function on $\mathcal{D}$, and $\mathrm{opt}_k(\mathcal{D}, \omega)$ is the maximum possible ω-weight of a set of at most k disjoint segments in $\mathcal{D}$; we may drop ω in the notation if the weight function is clear from the context. Also, whenever $\mathcal{D}$, ω, and k are clear from the context, then by an *optimum solution* we mean a set of pairwise disjoint segments $\mathcal{S} \subseteq \mathcal{D}$ such that $|\mathcal{S}| \leqslant k$ and $\omega(\mathcal{S}) = \mathrm{opt}_k(\mathcal{D})$.

4.1 Reducing the number of distinct weights

We first apply the same preprocessing as in Section 3: we guess a segment $R_{\max} \in \mathcal{S}$ of maximum weight and remove from $\mathcal{D}$ all segments of weight larger than $\omega(R_{\max})$ or not exceeding $\varepsilon \cdot \omega(R_{\max})/k$. Let $\mathcal{D}' \subseteq \mathcal{D}$ be the obtained set of segments. As argued in Section 3, we have

$$\mathrm{opt}_k(\mathcal{D}', \omega) \geqslant (1-\varepsilon) \cdot \mathrm{opt}_k(\mathcal{D}, \omega).$$

As the next preprocessing step, we round all weights down to the set

$$M \coloneqq \{\omega(R_{\max})(1-\varepsilon)^i \colon i \in \{0, 1, \ldots, \lceil \log_{1-\varepsilon}(\varepsilon/k) \rceil\}\}.$$

That is, we define the new weight function $\omega' \colon \mathcal{D}' \to \mathbb{R}$ by setting $\omega'(R)$ to be the largest element of M not exceeding $\omega(R)$. Since the weight of every segment is scaled down by a multiplicative factor of at most $1-\varepsilon$, we have

$$\mathrm{opt}_k(\mathcal{D}', \omega') \geqslant (1-\varepsilon) \cdot \mathrm{opt}_k(\mathcal{D}', \omega) \geqslant (1-\varepsilon)^2 \cdot \mathrm{opt}_k(\mathcal{D}, \omega) \geqslant (1-2\varepsilon) \cdot \mathrm{opt}_k(\mathcal{D}, \omega).$$

Thus, the two preprocessing steps above reduce solving the instance $(\mathcal{D}, \omega)$ to solving the instance $(\mathcal{D}', \omega')$, at the cost of losing $2\varepsilon \cdot \mathrm{opt}_k(\mathcal{D})$ on the optimum and a $|\mathcal{D}|^{\mathcal{O}(1)}$ overhead in the time complexity. Observe that in $(\mathcal{D}', \omega')$, the number of distinct weights of rectangles is bounded by $|M| \leqslant \mathcal{O}(\varepsilon \log(k/\varepsilon))$. We show in the sequel, that the MWISR problem for axis-parallel segments can be solved in fixed-parameter time when parameterized by both k and the number of distinct weights.

▶ **Theorem 4.1.** *Suppose $\mathcal{D}$ is a set of axis-parallel segments in the plane and ω is a weight function on $\mathcal{D}$. Let W be the number of distinct weights assigned by ω. Then given k, in time $(kW)^{\mathcal{O}(k^2)} \cdot |\mathcal{D}|^{\mathcal{O}(1)}$ one can find an optimum solution.*

Note that Theorem 1.2 follows from combining Theorem 4.1 with the preprocessing described above (applied to ε scaled by a factor of $1/2$). Therefore, from now on we focus on proving Theorem 4.1.

4.2 Constructing a grid

Let $\preceq$ be any total order on $\mathcal{D}$ that orders the segments by non-decreasing weights, that is, $\omega(R) < \omega(R')$ entails $R \prec R'$. Extend $\preceq$ to subsets of $\mathcal{D}$ in a lexicographic manner: $\mathcal{S} \prec \mathcal{S}'$ if $\mathcal{S}$ is lexicographically smaller than $\mathcal{S}'$ where both sets are sorted according to $\preceq$ (lexicographic order compares first the smallest objects). Let $\mathcal{S}_{\max}$ be the $\preceq$-maximum optimum solution.

The next step is to guess (by branching into $|\mathcal{D}|$ options) the $\preceq$-minimum segment $R_{\min}$ of $\mathcal{S}_{\max}$. Having done this, we safely remove from $\mathcal{D}$ all segments R satisfying $R \prec R_{\min}$. Since $\mathcal{S}_{\max}$ is the $\preceq$-maximum optimum solution, we achieve the following property: every optimum solution contains the $\preceq$-smallest segment of $\mathcal{D}$, i.e., $R_{\min}$. We proceed further with this assumption.

We now show that under this assumption, there must exist a grid of size at most k that hits every segment from $\mathcal{D}$. (Here and later on, a segment is *hit* by a line if they intersect, and a segment is *hit* by a grid if it is hit by a line in this grid.)

▷ Claim 4.2. Suppose every optimum solution contains the $\preceq$-minimum segment of $\mathcal{D}$. Then there exists a grid G of size at most k such that every segment in $\mathcal{D}$ is hit by G.

Proof. Let $R_{\min}$ be the $\preceq$-minimum segment of $\mathcal{D}$ and let $\mathcal{S}$ be any optimum solution. Let G be the grid comprising of, for every segment $R \in \mathcal{S}$, the line containing R. Clearly, we have $|G| \leqslant |\mathcal{S}| \leqslant k$. Suppose for contradiction, that there is a segment $R \in \mathcal{D}$ which is not hit by any line of G. Clearly $R \neq R_{\min}$, because $R_{\min} \in \mathcal{S}$ by assumption. Consider $\mathcal{S}' \coloneqq \mathcal{S} - \{R_0\} \cup \{R\}$ and note that $\mathcal{S}'$ is an independent set, because all segments of $\mathcal{S}$ are contained in lines of G, while R is disjoint with all those lines. Since $R_0 \prec R$, we have $\omega(R_0) \leqslant \omega(R)$, hence $\omega(\mathcal{S}') \geqslant \omega(\mathcal{S})$. So $\mathcal{S}'$ is an optimum solution that does not contain R_0, a contradiction. ◁

Note that the proof of Claim 4.2 is non-constructive, as the definition of the grid depends on the (unknown) optimum solution $\mathcal{S}$. However, we can give, following a greedy approach, a polynomial-time $\mathcal{O}(k)$-approximation algorithm for finding a grid that hits all segments in $\mathcal{D}$.

▶ **Lemma 4.3.** *There exists a polynomial time algorithm that given a set $\mathcal{D}$ of axis-parallel segments in the plane and an integer k, either correctly concludes that there is no grid of size at most k that hits all the segments of $\mathcal{D}$, or finds such a grid of size $\mathcal{O}(k^2)$.*

Proof of Lemma 4.3. We construct a grid G as follows. Swipe a vertical line from left to right across $\mathcal{D}$ until the first moment when the segments lying entirely to the left of the line can not be hit by k horizontal lines anymore. Let x_1 be the position of the line at this moment; in other words, x_1 is the least real such that the segments of $\mathcal{D}$ entirely contained in $(-\infty, x_1] \times \mathbb{R}$ cannot be hit with k horizontal lines. (We set $x_1 = \infty$ in case the whole $\mathcal{D}$ can be covered with at most k horizontal lines.) By the minimality of x_1, the segments entirely contained in $(-\infty, x_1] \times \mathbb{R}$ can be covered by $k+1$ lines: the k horizontal lines required to

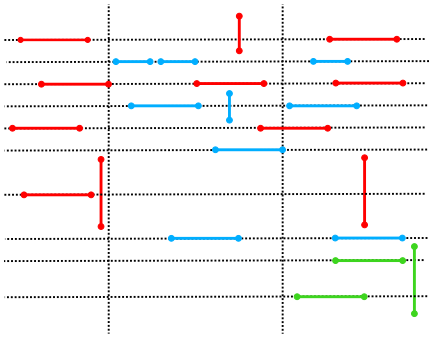

Figure 3 Example of the grid construction in the proof of Lemma 4.3, for $k = 3$. Red, blue, and green segments are removed in consecutive iterations. In each iteration we scan the segments from left to right until $k+1$ horizontal lines are needed to cover the already seen segments. In the last iteration at most k horizontal lines are selected. Lines added to G are dotted.

cover segments in $(-\infty, x_1)$, plus one vertical line at x_1 (in case $x_1 \neq \infty$). We add all those $k+1$ lines to G, delete from $\mathcal{D}$ all segments hit by those lines, and repeat the procedure until no more segments are left in $\mathcal{D}$. This way we obtain numbers $x_1 \leqslant x_2 \leqslant \ldots \leqslant x_\ell$ and a grid G of size at most $(k+1)\ell$, where ℓ is the number of iterations of the procedure. See Figure 3.

Clearly, G hits all segments in $\mathcal{D}$. So if $\ell \leqslant k+1$, then $|G| \leqslant (k+1)^2 = \mathcal{O}(k^2)$ and the algorithm can provide G as a valid output. We now argue that if $\ell > k+1$, then the algorithm may safely conclude that there is no grid of size at most k that hits all segments of $\mathcal{D}$. For contradiction, suppose there is such a grid G'. For $i \in \{1, \ldots, \ell\}$, let $\mathcal{D}_i$ be the set of all segments entirely contained in $(x_{i-1}, x_i] \times \mathbb{R}$, where we set $x_0 = -\infty$. It is easy to see that $\mathcal{D}_i$ is precisely the set of segments for which the algorithm in iteration i decided that it cannot be hit by at most k horizontal lines. Hence, for each $i \in \{1, \ldots, \ell\}$, G' must contain at least one vertical line hitting at least one segment in $\mathcal{D}_i$. The x-coordinate of this vertical line must belong to the interval $(x_{i-1}, x_i]$, so these vertical lines must be pairwise different. We conclude that $|G'| \geqslant \ell > k$, a contradiction.

It remains to argue how to implement the algorithm so that it runs in polynomial time. Observe that for a set of segments $\mathcal{D}' \subseteq \mathcal{D}$, the minimum number of horizontal lines needed to hit all the segments of $\mathcal{D}'$ can be computed as follows: Projecting all the segments $\mathcal{D}'$ on the vertical axis, and find the minimum number of points that hit the obtained set of intervals (some of which are single points; these are projected horizontal segments). This, in turn, can be done in time $\mathcal{O}(|\mathcal{D}'| \log |\mathcal{D}'|)$ using a standard greedy strategy. It is now straightforward to use this sub-procedure to execute the construction of G described above in polynomial time. ◀

We now combine Claim 4.2 and Lemma 4.3 as follows. Run the algorithm of Lemma 4.3 on $\mathcal{D}$ with parameter k. If the algorithm concludes that there is no grid of size at most k that hits all segments of $\mathcal{D}$, then by Claim 4.2 we can terminate the current branch, as clearly one of the previous guesses was incorrect. Otherwise, we obtain a grid G of size $\mathcal{O}(k^2)$ that hits every segment of $\mathcal{D}$. With this grid we proceed to the next steps.

For brevity of presentation, by adding four lines to G we may assume that all segments of $\mathcal{D}$ are contained in the interior of the rectangle delimited by the left-most and the right-most vertical line of G and the top-most and the bottom-most horizontal line of G. We will also say that a grid with this property *encloses* $\mathcal{D}$.

4.3 Constructing a nice grid

We use the same notion of niceness as in Section 3. That is, a grid G is *nice* with respect to a segment R, if R contains at least one grid point of G; in other words, R is intersected by both a horizontal and a vertical line in G. We will also say that R *respects* the grid G. The *ugliness* of a grid G with respect to some optimum solution $\mathcal{S}$ is the number of segments of $\mathcal{S}$ that do not respect G. Then the *ugliness* of G is the minimum over all optimum solutions $\mathcal{S}$ of the ugliness of G with respect to $\mathcal{S}$. This way, a grid is *nice* if its ugliness is 0, or equivalently, there exists an optimum solution $\mathcal{S}$ such that G is nice with respect to all the segments in $\mathcal{S}$.

In further considerations, it will be convenient to again rely on a suitable defined notion of a combinatorial type of a segment with respect to a grid. Consider a grid G that encloses $\mathcal{D}$. For a segment $R \in \mathcal{D}$, the *combinatorial type* of R with respect to G is the 6-tuple consisting of:

- The boolean value indicating whether R is horizontal or vertical.
- The weight $\omega(R)$.
- The right-most line $\ell_{\leftarrow}$ of G such that R entirely lies strictly to the right of $\ell_{\leftarrow}$.
- The left-most line $\ell_{\rightarrow}$ of G such that R entirely lies strictly to the left of $\ell_{\rightarrow}$.
- The bottom-most line $\ell_{\uparrow}$ of G such that R entirely lies strictly below $\ell_{\uparrow}$.
- The top-most line $\ell_{\downarrow}$ of G such that R entirely lies strictly above $\ell_{\downarrow}$.

In other words, $(\ell_{\leftarrow}, \ell_{\rightarrow}, \ell_{\uparrow}, \ell_{\downarrow})$ contain the sides of the inclusion-wise minimal rectangle R' delimited by the lines from G whose interior contains R. Note that the set of grid points of G contained in R is equal to the set of grid points contained in the interior of R'. Assuming G is clear from the context, for a type t we will denote this set of grid points by $P(t)$. Observe that the number of different combinatorial types with respect to G is bounded by $2W|G|^4$, where W is the number of distinct weights assigned by ω.

We first observe that if we manage to construct a small grid of ugliness 0, then we can find an optimum solution efficiently. The proof (in the appendix) boils down to guessing the combinatorial types of all segments and solving a suitable constructed instance of 2-CSP.

▶ **Lemma 4.4.** *Suppose we are given a finite set $\mathcal{D}$ of axis-parallel segments in the plane, a positive weight function ω on $\mathcal{D}$, a positive integer k, and a grid G that encloses $\mathcal{D}$ with a guarantee that the ugliness of G is 0. Then an optimum solution for $\mathcal{D}, \omega, k$ can be found in time $(W \cdot |G|)^{\mathcal{O}(k)} \cdot |\mathcal{D}|^{\mathcal{O}(1)}$.*

Proof. Fix any optimum solution $\mathcal{S}$ such that G is nice with respect to $\mathcal{S}$. We guess, by branching into all possibilities, the combinatorial types (with respect to G) of all the segments of $\mathcal{S}$. Since there are at most $2W|G|^4$ different combinatorial types, this results in $(W \cdot |G|)^{\mathcal{O}(k)}$ branches. Let the guessed set of combinatorial types be $\mathcal{T}$. Since G is supposed to be nice with respect to $\mathcal{S}$, we may assume that the sets $\{P(t) \colon t \in \mathcal{T}\}$ are nonempty and pairwise disjoint; otherwise the branch can be discarded.

We construct an auxiliary 2-CSP instance I that models the choice of segments in $\mathcal{S}$. The set of variables is $\mathcal{T}$. For every type $t \in \mathcal{T}$, the domain $\mathcal{D}_t$ consists of all segments from $\mathcal{D}$ whose combinatorial type is t. The constraints are as follows:

- If $t, t' \in \mathcal{T}$ are distinct types of horizontal segments, and $P(t)$ and $P(t')$ are two adjacent intervals of grid points on the same horizontal line of G, then we put a constraint between t and t' that among $\mathcal{D}_t \times \mathcal{D}_{t'}$, allows only pairs of disjoint segments.
- Analogous constraints are put for distinct types $t, t' \in \mathcal{T}$ of vertical segments for which $P(t)$ and $P(t')$ are adjacent intervals on the same vertical line.

It is straightforward to verify that solutions to I correspond in one-to-one fashion to those independent sets in $\mathcal{D}$ for which the set of combinatorial types is $\mathcal{T}$. Moreover, observe that the Gaifman graph of I is a disjoint union of paths, where every path $t_1 - \ldots - t_p$ corresponds

Figure 4 Illustration of Case 1 and Case 2 of the proof of Lemma 4.5. Red segments are the segments from $\mathcal{N}$ (they already contain a grid point). The green segment is the candidate segment $R_{\max}$ (maximum weight segment in the optimum solution not containing grid point). The box B is the blue-stroked rectangle. We greedily find a maximum-size region inside B containing candidate segments with the same combinatorial type as $R_{\max}$. If there are more than k independent candidates, we can return an optimum solution (since $R_{\max}$ has maximum weight). Otherwise, we can add fewer than k grid lines to the current grid (such that each candidate is hit by a newly added grid line).

to a sequence $P(t_1), \ldots, P(t_p)$ of intervals on the same grid line such that $P(t_i)$ is adjacent to $P(t_{i+1})$ for $i \in \{1, \ldots, p-1\}$. Therefore, it suffices to solve I optimally, which can be done in time $|\mathcal{D}|^{\mathcal{O}(1)}$ using, for instance, Theorem 2.1. ◀

Lemma 4.4 suggests that we should aim to construct a grid with zero ugliness. So far, the grid G constructed in the previous section may have positive ugliness: some segments of $\mathcal{D}$ may be intersected by just one, and not two orthogonal lines, and there is no reason why an optimum solution should not contain any such segments. Our goal is to reduce the ugliness of the grid by further branching steps. This strategy is captured in the following lemma.

▶ **Lemma 4.5 (★).** *Suppose we are given a finite set $\mathcal{D}$ of axis-parallel segments in the plane, a positive weight function ω on $\mathcal{D}$, a positive integer k, and a grid G that hits all segments of $\mathcal{D}$ and encloses $\mathcal{D}$. Let W be the number of different weights assigned by ω. Then one can construct, in time $(|G| \cdot W)^{\mathcal{O}(k)} \cdot |\mathcal{D}|^{\mathcal{O}(1)}$, a family $\mathcal{G}$ of grids with the following properties:*

1. *$|\mathcal{G}| \leqslant (|G| \cdot W)^{\mathcal{O}(k)}$;*
2. *for each $G' \in \mathcal{G}$, we have $G' \supseteq G$ and $|G' - G| \leqslant k$; and*
3. *If the ugliness of G is positive, then there is $G' \in \mathcal{G}$ whose ugliness is strictly smaller than that of G.*

While the complete proof of Lemma 4.5 can be found in the full version of this paper [10], we include here a sketch of the argument, because this is a crucial step.

Proof sketch of Lemma 4.5. Fix an optimum solution $\mathcal{S}$ that minimizes the ugliness of G. We assume that this ugliness is positive, otherwise we are done. The algorithm is a branching procedure that guesses some information about the structure of $\mathcal{S}$, and for each possible guess augments G so that at least one more segment of $\mathcal{S}$ respects G. Thus, different elements of $\mathcal{G}$ correspond to different guesses of the algorithm.

Let $\mathcal{N}$ consist of those segments in $\mathcal{S}$ that respect G. We guess the combinatorial types of all segments from $\mathcal{N}$ and, additionally, the heaviest segment $R_{\max}$ of $\mathcal{S} - \mathcal{N}$. As $R_{\max}$ does not respect G, it is easy to convince oneself that there are two possibilities of how $R_{\max}$ can be placed, depicted in Figure 4:

- $R_{\max}$ is not contained in any line of G, and thus is enclosed in the interior of a box B consisting of several consecutive cells of G.
- $R_{\max}$ is contained in a single segment of a line of G between two consecutive grid points.

The two cases are resolved using similar ideas. From now on we focus on the first case, and quickly comment on the second later.

Let $\mathcal{T}$ be the set of combinatorial types of segments from $\mathcal{N}$. Note that the algorithm does not know $\mathcal{N}$ (it is part of the solution), but has guessed $\mathcal{T}$. We first construct a set of segments $\mathcal{N}'$ with combinatorial types $\mathcal{T}$ (so with the same types as $\mathcal{N}$) that leaves "as much as possible" space within the box B for the placement of $R_{\max}$. Intuitively, this boils down to solving a suitable 2-CSP instance greedily by "pushing" candidate segments as much as possible away from B. Once $\mathcal{N}'$ is constructed, we look at all candidates for $R_{\max}$: segments with the guessed combinatorial type of $R_{\max}$. If these candidates lie on at least $k - |\mathcal{N}|$ different lines, then we actually found an optimal solution: we can output $\mathcal{N}'$ plus $k - |\mathcal{N}|$ disjoint candidates, each with the same weight as $R_{\max}$. Otherwise, all candidates lie on fewer than $k - |\mathcal{N}|$ lines, and we augment G by adding those lines. This ensures that, after augmentation, $R_{\max}$ respects G.

In the second case, we again greedily compute a set of segments $\mathcal{N}'$ with the same combinatorial types as $\mathcal{N}$ in such a way to leave as much space as possible on the segment between grid points where $R_{\max}$ is placed. Then we investigate the candidates for $R_{\max}$ and try to pack them greedily within the allowed space. If we are able to pack $k - |\mathcal{N}|$ of them, then we constructed an optimum solution consisting of $\mathcal{N}'$ plus this packing. Otherwise, we can hit all the candidates with fewer than $k - |\mathcal{N}|$ orthogonal grid lines; these are added to G ensuring that $R_{\max}$ respects G after augmentation. ◀

Finally, Lemma 4.5 can be applied in a recursive manner to obtain a nice grid.

▶ **Lemma 4.6.** *Suppose we are given a finite set $\mathcal{D}$ of axis-parallel segments in the plane, a positive weight function ω on $\mathcal{D}$, a positive integer k, and a grid G that hits all segments of $\mathcal{D}$ and encloses $\mathcal{D}$. Let W be the number of different weights assigned by ω. Then one can, in time $(k \cdot W \cdot |G|)^{\mathcal{O}(k^2)} \cdot |\mathcal{D}|^{\mathcal{O}(1)}$, construct a family $\mathcal{G}$ of grids such that:*

1. $|\mathcal{G}| \leqslant (k \cdot W \cdot |G|)^{\mathcal{O}(k^2)}$;
2. *for each $G' \in \mathcal{G}$, we have $G' \supseteq G$ and $|G' - G| \leqslant k^2$; and*
3. *$\mathcal{G}$ contains at least one grid of ugliness 0.*

Proof. Starting with $\mathcal{G}_0 \coloneqq \{G\}$, we iteratively construct $\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_k$ as follows: to construct $\mathcal{G}_i$ from $\mathcal{G}_{i-1}$, replace each $G \in \mathcal{G}_{i-1}$ with the family $\mathcal{G}(G)$ obtained by applying Lemma 4.5 to G. A straightforward induction using properties 1 and 2 of Lemma 4.5 shows that: $|\mathcal{G}_i| \leqslant (k \cdot W \cdot |G|)^{\mathcal{O}(ik)}$, for each $G' \in \mathcal{G}_i$ it holds that $G' \supseteq G$ and $|G' - G| \leqslant ik$, and the construction of $\mathcal{G}_i$ takes $(k \cdot W \cdot |G|)^{\mathcal{O}(ik)} \cdot |\mathcal{D}|^{\mathcal{O}(1)}$ time. Moreover, by property 3 of Lemma 4.5, if the minimum ugliness among grids in $\mathcal{G}_{i-1}$ is positive, then the minimum ugliness among the grids in $\mathcal{G}_i$ is strictly smaller than that in $\mathcal{G}_{i-1}$. Since the ugliness of G is at most k, it follows that $\mathcal{G} \coloneqq \mathcal{G}_k$ satisfies all the required properties. ◀

Theorem 4.1 is now a simple consequence of combining Lemma 4.6 with Lemma 4.4.

Proof of Theorem 4.1. As discussed in Section 4.2, by preprocessing the instance and branching into $\mathcal{O}(|\mathcal{D}|)$ possibilities, we may assume that we constructed a grid G of size $\mathcal{O}(k^2)$ such that every segment in $\mathcal{D}$ is hit by G. Adding four lines to G ensures that G encloses $\mathcal{D}$. Then we apply Lemma 4.6 to G, and in this manner we construct a family of grids $\mathcal{G}$ that features at least one grid with ugliness 0. It now remains to apply Lemma 4.4 to each grid in $\mathcal{G}$ and output the heaviest of the obtained solutions. Following directly from the guarantees provided by Lemmas 4.4 and 4.6, this algorithm runs in time $(kW)^{\mathcal{O}(k^2)} \cdot |\mathcal{D}|^{\mathcal{O}(1)}$. ◀

References

1 Anna Adamaszek and Andreas Wiese. Approximation Schemes for Maximum Weight Independent Set of Rectangles. In *IEEE 54th Annual Symposium on Foundations of Computer Science, FOCS 2013*, pages 400–409, 2013. doi:10.1109/FOCS.2013.50.

2 Pankaj K. Agarwal, Marc J. van Kreveld, and Subhash Suri. Label Placement by Maximum Independent Set in Rectangles. *Comput. Geom.*, 11(3-4):209–218, 1998. doi:10.1016/S0925-7721(98)00028-5.

3 Brenda S. Baker. Approximation algorithms for NP-complete problems on planar graphs. *J. ACM*, 41(1):153–180, 1994. doi:10.1145/174644.174650.

4 Cristina Bazgan. Schémas d'approximation et complexité paramétrée. *Rapport de stage de DEA d'Informatiquea Orsay*, 300, 1995. In French.

5 Paul S. Bonsma, Jens Schulz, and Andreas Wiese. A constant-factor approximation algorithm for unsplittable flow on paths. *SIAM J. Comput.*, 43(2):767–799, 2014. doi:10.1137/120868360.

6 Clément Carbonnel, Miguel Romero, and Stanislav Živný. Point-Width and Max-CSPs. *ACM Trans. Algorithms*, 16(4):54:1–54:28, 2020. doi:10.1145/3409447.

7 Marco Cesati and Luca Trevisan. On the efficiency of polynomial time approximation schemes. *Information Processing Letters*, 64(4):165–171, 1997. doi:10.1016/S0020-0190(97)00164-6.

8 Parinya Chalermsook and Bartosz Walczak. Coloring and Maximum Weight Independent Set of Rectangles. In *2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021*, pages 860–868, 2021. doi:10.1137/1.9781611976465.54.

9 Julia Chuzhoy and Alina Ene. On Approximating Maximum Independent Set of Rectangles. In *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016*, pages 820–829, 2016. doi:10.1109/FOCS.2016.92.

10 Jana Cslovjecsek, Michal Pilipczuk, and Karol Wegrzycki. Parameterized approximation for maximum weight independent set of rectangles and segments. *CoRR*, abs/2212.01620, 2022. doi:10.48550/arXiv.2212.01620.

11 Jana Cslovjecsek, Michal Pilipczuk, and Karol Węgrzycki. A polynomial-time $\mathrm{OPT}^{\varepsilon}$-approximation algorithm for maximum independent set of connected subgraphs in a planar graph. In *Proceedings of the 2024 ACM-SIAM Symposium on Discrete Algorithms, SODA 2024*, pages 625–638. SIAM, 2024.

12 Jeffrey S. Doerschler and Herbert Freeman. A rule-based system for dense-map name placement. *Commun. ACM*, 35(1):68–79, 1992. doi:10.1145/129617.129620.

13 Thomas Erlebach, Klaus Jansen, and Eike Seidel. Polynomial-Time Approximation Schemes for Geometric Intersection Graphs. *SIAM J. Comput.*, 34(6):1302–1323, 2005. doi:10.1137/S0097539702402676.

14 Robert J. Fowler, Michael S. Paterson, and Steven L. Tanimoto. Optimal packing and covering in the plane are NP-complete. *Information Processing Letters*, 12(3):133–137, 1981. doi:10.1016/0020-0190(81)90111-3.

15 Eugene C. Freuder. Complexity of k-tree structured constraint satisfaction problems. In *8th National Conference on Artificial Intelligence*, pages 4–9, 1990. URL: http://www.aaai.org/Library/AAAI/1990/aaai90-001.php.

16 Takeshi Fukuda, Yasuhiko Morimoto, Shinichi Morishita, and Takeshi Tokuyama. Data mining with optimized two-dimensional association rules. *ACM Trans. Database Syst.*, 26(2):179–213, 2001. URL: http://portal.acm.org/citation.cfm?id=383891.383893, doi:10.1145/383891.383893.

17 Waldo Gálvez, Arindam Khan, Mathieu Mari, Tobias Mömke, Madhusudhan Reddy Pittu, and Andreas Wiese. A 3-Approximation Algorithm for Maximum Independent Set of Rectangles. In *2022 ACM-SIAM Symposium on Discrete Algorithms, SODA 2022*, pages 894–905, 2022. doi:10.1137/1.9781611977073.38.

18 Fabrizio Grandoni, Edin Husić, Mathieu Mari, and Antoine Tinguely. Approximating the Maximum Independent Set of Convex Polygons with a Bounded Number of Directions. *Accepted to SoCG 2024*, 2024.

19 Fabrizio Grandoni, Stefan Kratsch, and Andreas Wiese. Parameterized approximation schemes for Independent Set of Rectangles and Geometric Knapsack. In *27th Annual European Symposium on Algorithms, ESA 2019*, volume 144 of *LIPIcs*, pages 53:1–53:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.ESA.2019.53.

20 Jan Kára and Jan Kratochvíl. Fixed Parameter Tractability of Independent Set in Segment Intersection Graphs. In *Second International Workshop on Parameterized and Exact Computation, IWPEC 2006*, pages 166–174, 2006. doi:10.1007/11847250_15.

21 Liane Lewin-Eytan, Joseph Naor, and Ariel Orda. Routing and admission control in networks with advance reservations. In *5th International Workshop on Approximation Algorithms for Combinatorial Optimization, APPROX 2002*, pages 215–228, 2002. doi:10.1007/3-540-45753-4_19.

22 Dániel Marx. Efficient approximation schemes for geometric problems? In *13th Annual European Symposium on Algorithms, ESA 2005*, pages 448–459, 2005. doi:10.1007/11561071_41.

23 Dániel Marx. Parameterized Complexity of Independence and Domination on Geometric Graphs. In *Second International Workshop on Parameterized and Exact Computation, IWPEC 2006*, pages 154–165, 2006. doi:10.1007/11847250_14.

24 Joseph S. B. Mitchell. Approximating Maximum Independent Set for rectangles in the plane. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021*, pages 339–350, 2021. doi:10.1109/FOCS52979.2021.00042.

25 Neil Robertson and Paul D. Seymour. Graph minors. XIII. The disjoint paths problem. *J. Comb. Theory, Ser. B*, 63(1):65–110, 1995. doi:10.1006/jctb.1995.1006.

26 Miguel Romero, Marcin Wrochna, and Stanislav Živný. Treewidth-Pliability and PTAS for Max-CSPs. In *2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021*, pages 473–483, 2021. doi:10.1137/1.9781611976465.29.