Approximation and Parameterized Algorithms for Geometric Independent Set with Shrinking*

Michał Pilipczuk[†1], **Erik Jan van Leeuwen**[2], **and Andreas Wiese**[3]

1 **Institute of Informatics, University of Warsaw, Poland**
michal.pilipczuk@mimuw.edu.pl
2 **Department of Information and Computing Sciences, Utrecht University, The Netherlands**
e.j.vanleeuwen@uu.nl
3 **Department of Industrial Engineering and Center for Mathematical Modeling, Universidad de Chile, Chile**
awiese@dii.uchile.cl

Abstract

Consider the Maximum Weight Independent Set problem for rectangles: given a family of weighted axis-parallel rectangles in the plane, find a maximum-weight subset of non-overlapping rectangles. The problem is notoriously hard both in the approximation and in the parameterized setting. The best known polynomial-time approximation algorithms achieve super-constant approximation ratios [5, 7], even though there is a $(1+\epsilon)$-approximation running in quasi-polynomial time [2, 8]. When parameterized by the target size of the solution, the problem is W[1]-hard even in the unweighted setting [12].

To achieve tractability, we study the following *shrinking model*: one is allowed to shrink each input rectangle by a multiplicative factor $1-\delta$ for some fixed $\delta > 0$, but the performance is still compared against the optimal solution for the original, non-shrunk instance. We prove that in this regime, the problem admits an EPTAS with running time $f(\epsilon,\delta)\cdot n^{\mathcal{O}(1)}$, and an FPT algorithm with running time $f(k,\delta)\cdot n^{\mathcal{O}(1)}$, in the setting where a maximum-weight solution of size at most k is to be computed. This improves and significantly simplifies a PTAS given earlier for this problem [1], and provides the first parameterized results for the shrinking model. Furthermore, we explore kernelization in the shrinking model, by giving efficient kernelization procedures for several variants of the problem when the input rectangles are squares.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases Combinatorial optimization, Approximation algorithms, Fixed-parameter algorithms

Digital Object Identifier 10.4230/LIPIcs.MFCS.2017.42

1 Introduction

In Maximum (Weight) Independent Set, given a graph, the goal is to select a set of pairwise non-adjacent vertices with maximum cardinality or total weight. In its full generality, the problem is NP-hard and intractable both in the approximation and in the parameterized

* A full version of the paper is available at https://arxiv.org/abs/1611.06501.
† The research of Mi. Pilipczuk is supported by Polish National Science Centre grant UMO-2013/11/D/ST6/03073. Mi. Pilipczuk is also supported by the Foundation for Polish Science (FNP) via the START stipend programme.

setting: it is NP-hard to approximate within ratio $n^{1-\epsilon}$ for any $\epsilon > 0$ [16], and it is W[1]-hard when parameterized by the solution size [9]. Therefore, many restricted settings were studied.

One well-studied case is to consider a geometric setting where the input consists of a family of geometric objects, and the goal is to select a maximum-weight subfamily of pairwise non-overlapping objects. This case reduces to the graph setting by considering the intersection graph of the objects. These graphs are highly structured, which gives hope for better results than for general graphs.

This paper concentrates on the variant in which the input objects are axis-parallel rectangles in the two-dimensional plane. In this variant, Maximum Weight Independent Set admits much smaller approximation ratios than on general graphs. While no polynomial-time constant-factor approximation algorithm is known in general, there is an $\mathcal{O}(\log \log n)$-approximation algorithm for unweighted rectangles [5], an $\mathcal{O}(\log n / \log \log n)$-approximation algorithm for weighted rectangles [7], and a PTAS for squares [6, 10]. If one allows quasi-polynomial running time, then there is a $(1+\epsilon)$-approximation algorithm (a *QPTAS*) [3, 11]. It remains open whether this can be improved to a PTAS.

From the parameterized perspective, the problem remains W[1]-hard when parameterized by the size k of the solution, even for unweighted unit squares [12]. Therefore, the existence of an FPT algorithm with running time $f(k) \cdot n^{\mathcal{O}(1)}$ for a computable f is unlikely under standard assumptions from parameterized complexity; this also excludes the existence of an EPTAS for the problem [12]. However, the problem admits a faster-than-brute-force parameterized algorithm with running time $n^{\mathcal{O}(\sqrt{k})}$, which is optimal under the Exponential Time Hypothesis [13]. This algorithm works in the general setting of finding a maximum-weight independent set of size k in a family of polygons in the plane.

Shrinking model. In order to circumvent some of the many challenges that arise when designing approximation or parameterized algorithms for geometric Maximum Weight Independent Set, we investigate the *shrinking model* introduced by Adamaszek et al. [1]. In this model, one is allowed to shrink each input object by a multiplicative factor $1-\delta$ for some fixed $\delta > 0$, but the weight of the computed solution is still compared to the optimum for the original, non-shrunk instance; we give a formal definition later. It is known that the shrinking model allows for substantially better approximation algorithms than the general setting: Adamaszek et al. [1] gave a PTAS for axis-parallel rectangles, which was later generalized by Wiese to arbitrary convex polygons [15]. However, it has not been studied so far whether shrinking also helps to design parameterized algorithms. One concrete question would be whether Independent Set for axis-parallel rectangles remains W[1]-hard in the shrinking model.

Our results. This paper addresses the parameterized complexity of Maximum Weight Independent Set of Rectangles in the shrinking model, and answers the above questions. On the way to our two main parameterized contributions, we also improve the PTAS by Adamaszek et al. [1] to an EPTAS.

Our first main contribution is that Maximum Weight Independent Set of Rectangles is fixed-parameter tractable (FPT) in the shrinking model. Formally, for a shrinking parameter δ, we can decide in (deterministic) time $f(k,\delta) \cdot (nN)^{\mathcal{O}(1)}$ whether there is an independent set of k (shrunk) rectangles, or the original family has no independent subfamily of size k. Here, N is the total bit size of the input and f is some computable function. The algorithm also works in the weighted setting, where we look for a maximum-weight subset of at most k non-overlapping rectangles. The reason why we are able to circumvent

the W[1]-hardness for the standard model (i.e., without shrinking) is that the reduction of Marx [12] relies on tiny differences in the coordinates of the rectangles. However, as Adamaszek et al. [1] and this paper show, this aspect vanishes in the shrinking model.

The parameterized algorithm is actually a consequence of an EPTAS that we present for Maximum Weight Independent Set of Rectangles. That is, we give an algorithm with running time $f(\epsilon,\delta)\cdot(nN)^{\mathcal{O}(1)}$ that finds a subset of rectangles that do not overlap after shrinking by factor $1-\delta$, and whose total weight is at least $1-\epsilon$ times the optimum without shrinking. Recall that the standard model does not admit an EPTAS, unless FPT = W[1] [12].

Our EPTAS is based on the same principles as the PTAS of Adamaszek et al. [1]. The idea is to assemble an optimum solution using a bottom-up dynamic-programming approach pioneered by Erlebach et al. [10]. Each subproblem solved in the dynamic program corresponds to the maximum weight of an independent set contained in a "box", and the computation of the optimum for each such box boils down to enumerating a limited number of carefully chosen partitions of the box into smaller boxes. Intuitively, the ability to shrink is used to make sure that rectangles fit nicely into the different boxes. The main challenge is to ensure that these boxes can be assumed to be simple, and therefore only a limited number of subproblems is necessary to assemble a near-optimum solution.

The crucial contribution in our approximation algorithm is that we show that rectangular boxes suffice. In [1], most rectangles were shrunk in *only one direction* and therefore, the boxes were axis-parallel polygons with at most $g(\epsilon,\delta)$ sides each, for some function g. This makes the dynamic program very complex, and yields a running time of $(nN)^{g(\epsilon,\delta)}$ due to the sheer number of subproblems solved. In this paper, we fully exploit the properties of the shrinking model and shrink *each* rectangle in *two directions*. This changes the analysis, but the main advantage is that we only need to consider boxes that are rectangles (i.e., with only four sides) in our dynamic program. This greatly simplifies the dynamic program, and we show that we need to consider only $f(\epsilon,\delta)\cdot(nN)^{\mathcal{O}(1)}$ different subproblems. Hence, our EPTAS is both substantially faster and significantly simpler than previous work.

Our second main contribution is showing that several important subcases of Maximum Weight Independent Set of Rectangles with δ-shrinking admit polynomial kernels when parameterized by k and δ. Intuitively, such a kernel is a polynomial-time computable subfamily of the input rectangles of size bounded by a polynomial of k and δ that retains an optimum solution after δ-shrinking; a formal definition is given in Section 4. For unit squares of non-uniform weight, we construct a kernel of size $\mathcal{O}(k/\delta^2)$, while for squares of non-uniform size, but of uniform weight, we construct a kernel of size $\mathcal{O}(k^2\cdot\frac{\log(1/\delta)}{\delta^3})$. As a direct consequence, we obtain FPT algorithms for the considered variants with running time $(k/\delta)^{\mathcal{O}(\sqrt{k})}\cdot(nN)^{\mathcal{O}(1)}$ by applying the $n^{\mathcal{O}(\sqrt{k})}$-time algorithm of Marx and Pilipczuk [13] on the kernels. This *subexponential* running time is far better than the running time of our FPT algorithm for the general case.

Organization. In this extended abstract we sketch our EPTAS and present the main ideas behind the kernelization results. A much broader discussion, including complete proofs of all results, can be found in the full version available on the arXiv [14]. The proofs of claims marked with ♠ appear in the full version [14].

2 Preliminaries

We essentially adopt the notation of Adamaszek et al. [1]. Suppose that $\mathcal{R}=\{R_1,R_2,\ldots,R_n\}$ is a family of axis-parallel rectangles given in the input. Each rectangle R_i is described as $R_i=\{(a,b)\,:\,x_i^{(1)}<a<x_i^{(2)}$ and $y_i^{(1)}<b<y_i^{(2)}\}$, where $x_i^{(1)}<x_i^{(2)}$ and $y_i^{(1)}<y_i^{(2)}$

are integers. Thus, the input rectangles are assumed to be open, and their vertices are at integral points. We assume that the family $\mathcal{R}$ is given in the input with all the coordinates $x_i^{(1)}, x_i^{(2)}, y_i^{(1)}, y_i^{(2)}$ encoded in binary; thus, the coordinates are at most exponential in the total bit size of the input, denoted by N. For a rectangle R_i, we define its *width* $g_i = x_i^{(2)} - x_i^{(1)}$ and *height* $h_i = y_i^{(2)} - y_i^{(1)}$. Moreover, each rectangle R_i has an associated *weight* w_i, which is a nonnegative real. For a subset $\mathcal{S} \subseteq \mathcal{R}$, we denote $w(\mathcal{S}) = \sum_{R_i \in \mathcal{S}} w_i$.

Fix a constant δ with $0 < \delta < 1$. For a rectangle R_i, its *δ-shrinking* $R_i^{-\delta}$ is the rectangle with x-coordinates $x_i^{(1)} + \frac{\delta}{2} g_i$ and $x_i^{(2)} - \frac{\delta}{2} g_i$, and y-coordinates $y_i^{(1)} + \frac{\delta}{2} h_i$ and $y_i^{(2)} - \frac{\delta}{2} h_i$. The δ-shrinking retains the weight w_i of the original rectangle. For a subset $\mathcal{S} \subseteq \mathcal{R}$, we denote $\mathcal{S}^{-\delta} = \{R_i^{-\delta} \colon R_i \in \mathcal{S}\}$ to be the family of δ-shrinkings of rectangles from $\mathcal{S}$.

A family of rectangles is *independent* (or is an *independent set*) if the rectangles are pairwise non-overlapping. In the Maximum Weight Independent Set of Rectangles problem (MWISR) we are given a family of axis-parallel rectangles $\mathcal{R} = \{R_1, R_2, \ldots, R_n\}$, and the goal is to find a subfamily of $\mathcal{R}$ that is independent and has maximum total weight. This maximum weight will be denoted by $\mathsf{OPT}(\mathcal{R})$. In the parameterized setting, we are additionally given an integer parameter k, and we look for a subfamily of $\mathcal{R}$ that has size at most k, is independent, and has maximum possible weight subject to these conditions. This maximum weight will be denoted by $\mathsf{OPT}_k(\mathcal{R})$.

In the δ-shrinking setting, we relax the requirement of independence to just requiring the disjointness of δ-shrinkings, but we still compare the weight of the output of our algorithm with $\mathsf{OPT}(\mathcal{R})$, respectively with $\mathsf{OPT}_k(\mathcal{R})$.

3 Main results

With the above definitions in mind, we can state formally our main results.

▶ **Theorem 1** (FPT for MWISR with δ-shrinking). *There is a deterministic algorithm that, given a weighted family $\mathcal{R}$ of n axis-parallel rectangles with total encoding size N and parameters k and δ, runs in time $f(k,\delta) \cdot (nN)^c$ for some computable function f and constant c, and outputs a subfamily $\mathcal{S} \subseteq \mathcal{R}$ such that $|\mathcal{S}| \leq k$, $\mathcal{S}^{-\delta}$ is independent, and $w(\mathcal{S}) \geq \mathsf{OPT}_k(\mathcal{R})$.*

▶ **Theorem 2** (EPTAS for MWISR with δ-shrinking). *There is a deterministic algorithm that, given a weighted family $\mathcal{R}$ of n axis-parallel rectangles with total encoding size N and parameters δ, ϵ, runs in time $f(\epsilon,\delta) \cdot (nN)^c$ for some computable function f and constant c, and outputs a subfamily $\mathcal{S} \subseteq \mathcal{R}$ such that $\mathcal{S}^{-\delta}$ is independent, and $w(\mathcal{S}) \geq (1-\epsilon)\mathsf{OPT}(\mathcal{R})$.*

In this section we sketch the proof of Theorem 2. Theorem 1 follows by a simple adjustment of the reasoning, as explained in the full version of the paper [14].

Throughout the proof we fix the input family $\mathcal{R} = \{R_1, \ldots, R_n\}$, and we denote $\mathsf{OPT}(\mathcal{R})$ by OPT. We also fix the constants δ and ϵ, and w.l.o.g. we assume that $1/\delta$ and $1/\epsilon$ are even integers larger than 4. For convenience, throughout the proof we aim at finding a solution $\mathcal{S}$ with $w(\mathcal{S}) \geq (1 - d \cdot \epsilon)\mathsf{OPT}$ for some constant d, for at the end we may rescale the parameter ϵ to ϵ/d. By shifting all the rectangles, we may assume without loss of generality that they all fit into the square $[1, L] \times [1, L]$, where $L = (1/\delta\epsilon)^\ell$ for some integer $\ell = \mathcal{O}(N)$. That is, all the coordinates $x_i^{(1)}, x_i^{(2)}, y_i^{(1)}, y_i^{(2)}$ are between 1 and L, so in particular the width and the height of each rectangle is smaller than L.

We divide our reasoning into two steps. First, like in the PTAS of Adamaszek et al. [1], in Section 3.1 we describe how to remove some rectangles from $\mathcal{R}$ using standard shifting arguments so that OPT decreases only by an $\mathcal{O}(\epsilon)$-fraction, but the resulting family admits

some useful properties. Then, we shrink the rectangles in a similar way as in [1]; however, in contrast to [1] we will shrink *each* rectangle in both directions which will be important in our analysis. Second, we show that the properties of the obtained family enable us to compute an optimum solution using dynamic programming; this algorithm is presented in Section 3.2.

3.1 Sparsifying the family

Intuitively, we will apply shifting techniques to extract some structure in the input family $\mathcal{R}$ while losing only an $\mathcal{O}(\epsilon)$-fraction of OPT. The first goal is to classify the rectangles according to their widths (respectively, heights) such that rectangles in the same class have similar widths (respectively, heights), but between the classes the dimensions differ significantly.

▶ **Definition 3.** A subfamily $\mathcal{R}' \subseteq \mathcal{R}$ is *well-separated* if there exist two partitions $(\mathcal{R}^{\mathsf{V}}_1, \ldots, \mathcal{R}^{\mathsf{V}}_p)$ and $(\mathcal{R}^{\mathsf{H}}_1, \ldots, \mathcal{R}^{\mathsf{H}}_p)$ of $\mathcal{R}'$, with $p \leq \ell$, as well as reals ν_t, μ_t for $t = 1, 2, \ldots, p$, with the following properties satisfied for each $t \in \{1, 2, \ldots, p\}$:

- $\nu_t \leq g_i < \mu_t$ for each $R_i \in \mathcal{R}^{\mathsf{V}}_t$;
- $\nu_t \leq h_i < \mu_t$ for each $R_i \in \mathcal{R}^{\mathsf{H}}_t$;
- $\nu_t / \mu_{t-1} = 1/\delta\epsilon$ (except for $t = 1$) and $\mu_t / \nu_t = (1/\delta\epsilon)^{(1/\epsilon)-1}$; and
- $\nu_1 \leq 1$, $\mu_p \geq L$, and all numbers ν_t and μ_t apart from ν_1 are integers.

The partitions $(\mathcal{R}^{\mathsf{V}}_t)_{t=1,\ldots,p}$ and $(\mathcal{R}^{\mathsf{H}}_t)_{t=1,\ldots,p}$ are called the *vertical* and *horizontal levels*, respectively, whereas the parameters $(\nu_t)_{t=1,\ldots,p}$ and $(\mu_t)_{t=1,\ldots,p}$ are the *lower* and *upper limits* of the corresponding levels. Note that vertical levels partition $\mathcal{R}'$ by width, while the horizontal levels partition $\mathcal{R}'$ by height.

We now prove that we can find a well-separated subfamily that loses only an $\mathcal{O}(\epsilon)$-fraction of OPT using a standard shifting technique. Essentially the same step is used in the PTAS of Adamaszek et al. [1] (see Lemma 6 therein). Henceforth we will use the notation $[q] = \{0, 1, \ldots, q-1\}$ for any positive integer q.

▶ **Lemma 4.** *We can compute a collection of $1/\epsilon$ subfamilies $\mathcal{R}'_0, \ldots, \mathcal{R}'_{1/\epsilon-1} \subseteq \mathcal{R}$ in polynomial time such that each subfamily is well-separated, and there exists a $b^* \in [1/\epsilon]$ for which $\mathsf{OPT}(\mathcal{R}'_{b^*}) \geq (1 - 2\epsilon)\mathsf{OPT}$.*

Proof. Recall that the widths and heights of the rectangles from $\mathcal{R}$ are integers between 1 and $L-1$, where $L = (1/\delta\epsilon)^\ell$. First, create a partition of the rectangles into *vertical layers* $\mathcal{L}^{\mathsf{V}}_j$ for $j = 1, 2, \ldots, \ell$, where layer $\mathcal{L}^{\mathsf{V}}_j$ consists of rectangles R_i for which $(1/\delta\epsilon)^{j-1} \leq g_i < (1/\delta\epsilon)^j$. In a symmetric manner, partition $\mathcal{R}$ into *horizontal layers* $\mathcal{L}^{\mathsf{H}}_j$ for $j = 1, 2, \ldots, \ell$, where layer $\mathcal{L}^{\mathsf{H}}_j$ consists of rectangles R_i for which $(1/\delta\epsilon)^{j-1} \leq h_i < (1/\delta\epsilon)^j$.

For each offset $b \in [1/\epsilon]$ we construct a subfamily $\mathcal{R}'_b$ from $\mathcal{R}$ by removing all rectangles contained in those vertical layers $\mathcal{L}^{\mathsf{V}}_j$ and those horizontal layers $\mathcal{L}^{\mathsf{H}}_j$, for which $j \equiv b \mod (1/\epsilon)$. It is easy to see that each subfamily $\mathcal{R}'_b$ constructed in this manner is well-separated: each vertical level $\mathcal{R}^{\mathsf{V}}_t \subseteq \mathcal{R}'_b$ consists of $(1/\epsilon) - 1$ consecutive vertical layers between two removed ones, with the exception of the first and the last level, for which the start/end of the sequence of layers delimits the level. A symmetric analysis yields the partition into horizontal levels. It is straightforward to compute in polynomial time the partition into horizontal/vertical levels, as well as to choose their lower and upper limits.

Suppose that we choose b uniformly at random from the set $[1/\epsilon]$. Fix any optimum solution $\mathcal{S}$ in $\mathcal{R}$, that is, an independent set of rectangles such that $w(\mathcal{S}) = \mathsf{OPT}$. Observe that for any rectangle $R_i \in \mathcal{S}$, the probability that the vertical layer it belongs to is removed during the construction of $\mathcal{R}'_b$, is equal to ϵ. Similarly, the probability that the horizontal layer to which R_i belongs is removed when constructing $\mathcal{R}'_b$, is also ϵ. Hence, R_i is not

included in $\mathcal{R}'_b$ with probability at most 2ϵ. This means that the expected value of $w(\mathcal{S} \setminus \mathcal{R}'_b)$, the total weight of rectangles from $\mathcal{S}$ that did not survive in $\mathcal{R}'_b$, is at most $2\epsilon \cdot \mathsf{OPT}$. Hence, in expectation we have that $w(\mathcal{S} \setminus \mathcal{R}'_b) \leq 2\epsilon \cdot \mathsf{OPT}$. Therefore, there exists a subfamily $\mathcal{R}'_{b^*}$ with $b^* \in [1/\epsilon]$ such that $\mathsf{OPT}(\mathcal{R}'_{b^*}) \geq (1 - 2\epsilon)\mathsf{OPT}$. ◀

We now execute the rest of the algorithm on $\mathcal{R}'_b$ for each $b \in [1/\epsilon]$, and output the best solution obtained overall. This increases the running time of the algorithm by a factor $1/\epsilon$. From Lemma 4, however, we know that $\mathsf{OPT}(\mathcal{R}'_b) \geq (1 - 2\epsilon)\mathsf{OPT}$ for $b = b^*$, and thus we lose at most a factor $(1 - 2\epsilon)$ in this way. From now on, let $\mathcal{R}' = \mathcal{R}'_b$ for some $b \in [1/\epsilon]$.

Hierarchical grid structure. For any integer a, we describe a hierarchical grid structure and remove rectangles along it. We then execute the rest of the algorithm for a bounded number of values of a, losing at most an additional factor $(1 - 6\epsilon)$ in this way. Given a value of a, the grid structure is constructed as follows. We first divide the horizontal lines into p levels (p as in Definition 3) corresponding to the horizontal levels $\mathcal{R}^{\mathsf{H}}_t$. For level t, define the *level-t unit* as $u_t = \delta\nu_t/2$. Thus, for $t > 1$, we have $u_t = \mu_{t-1}/(2\epsilon)$, and hence u_t is an integer for $t > 1$, since $1/\epsilon$ is even. For each level $t \in \{1, 2, \ldots, p\}$ we define a set of horizontal grid lines G^{H}_t, consisting of the horizontal lines with y-coordinates from the set $\{a + b \cdot u_t \colon b \in \mathbb{Z}\}$. In other words, we take horizontal lines that are u_t apart from each other, and we shift them so that there is a line with y-coordinate a. We define vertical grid lines G^{V}_t of levels $t = 1, 2, \ldots, p$ in a symmetric manner, using the same shift parameter a and the same units for all levels. Define the *grid* of level t to be $G_t = G^{\mathsf{H}}_t \cup G^{\mathsf{V}}_t$. Note that any line of G_t is also a line of $G_{t'}$ for all $t' < t$. Thus, the grid of each level t' refines the grid of each larger level t.

Before we proceed, we describe the intuition of the next step; this step is also present in the PTAS of Adamaszek et al. [1] (see Lemma 7 therein). Rectangles belonging to $\mathcal{R}^{\mathsf{V}}_{t'}$ for $t' \geq t$ have width not smaller than ν_t. On the other hand, the lines of G^{V}_t are spaced at distance $u_t = \delta\nu_t/2$ apart, which means that there are $\Omega(1/\delta)$ vertical grid lines of G^{V}_t crossing each rectangle of vertical level t or larger. Intuitively, G^{V}_t provides a fine grid for those vertical levels, so that their rectangles can be snapped to the lines of G^{V}_t via shrinking by a multiplicative factor of at most $1 - \delta$. On the other hand, the rectangles of vertical levels $t - 1$ or smaller have widths not larger than $\mu_{t+1} = \nu_t \cdot (\delta\epsilon)$. Hence, the grid lines of G^{V}_t are at much larger distance to each other than the maximum possible width of such rectangles; more precisely, larger by a multiplicative factor at least $1/(2\epsilon)$. Consequently, if we were to choose $a \in [L]$ uniformly at random, then the probability that a rectangle R_i will be crossed by a vertical line of level larger than its vertical level, or a horizontal line of level larger than its horizontal level, will be $\mathcal{O}(\epsilon)$. If we exclude such rectangles, then we lose only an $\mathcal{O}(\epsilon)$-fraction of OPT in expectation. We now formalize the above intuition and use it to derive an existential statement and a deterministic algorithm.

We need the following definition. Let $R_i \in \mathcal{R}'$, and suppose that $R_i \in \mathcal{R}^{\mathsf{V}}_s \cap \mathcal{R}^{\mathsf{H}}_t$. We say that R_i is *abusive* if R_i is crossed by a vertical line of level larger than s, or R_i is crossed by a horizontal line of level larger than t; see Figure 1. A family of rectangles without abusive rectangles (with respect to the hierarchical grid structure for a) is called *well-behaved (for a)*.

▶ **Lemma 5.** *Let $U := \sum_{t=1}^{p} u_t$. We can compute a collection of $(1/\delta\epsilon)^{1/\epsilon}$ subfamilies $\mathcal{R}''_0, \ldots, \mathcal{R}''_{(1/\delta\epsilon)^{1/\epsilon}-1} \subseteq \mathcal{R}'$ in $(1/\delta\epsilon)^{1/\epsilon} \cdot (nN)^{\mathcal{O}(1)}$ time such that, for each $c \in [(1/\delta\epsilon)^{1/\epsilon}]$, the subfamily $\mathcal{R}''_c$ is well-separated and well-behaved for $c \cdot U$, and there exists a $c^* \in [(1/\delta\epsilon)^{1/\epsilon}]$ for which $\mathsf{OPT}(\mathcal{R}''_{c^*}) \geq (1 - 6\epsilon)\mathsf{OPT}(\mathcal{R}')$.*

Proof. For each $c \in [(1/\delta\epsilon)^{1/\epsilon}]$, the family $\mathcal{R}''_c$ is obtained from $\mathcal{R}'$ by removing all rectangles that are abusive with respect to the hierarchical grid structure for $a = c \cdot U$. Hence, $\mathcal{R}''_c$ is well-behaved for $c \cdot U$. Since we are only removing rectangles, $\mathcal{R}''_c$ is still well-separated.

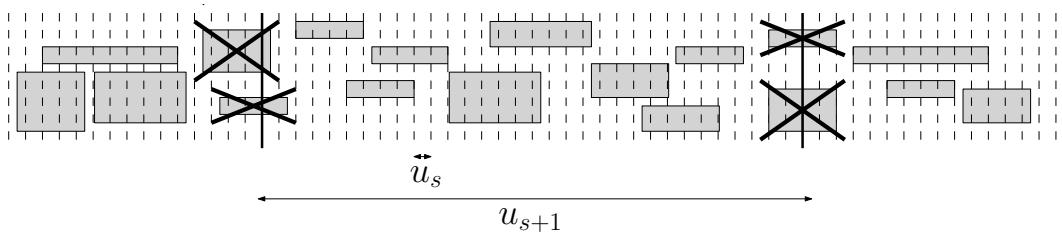

Figure 1 The vertical grid. The dashed vertical lines are the vertical grid lines of G_s^{V}, the bold vertical lines are the lines in the set G_{s+1}^{V}. All shown rectangles are from level $\mathcal{R}_s^{\mathsf{V}}$. The crossed out rectangles are abusive since they intersect lines from G_{s+1}^{V}.

It remains to show the existence of c^*. Let $R_i \in \mathcal{R}_s^{\mathsf{V}}$. As $R_i \in \mathcal{R}_s^{\mathsf{V}}$, we have that $g_i < \mu_s = \nu_{s+1} \cdot (\delta\epsilon) = u_{s+1} \cdot 2\epsilon$. Hence,

$$\frac{g_i}{\sum_{r=1}^{s} u_r} \leq \frac{g_i}{u_s} < \frac{u_{s+1} \cdot 2\epsilon}{u_s} = 2\epsilon \cdot (1/\delta\epsilon)^{1/\epsilon}. \tag{1}$$

Note that this inequality holds regardless of the choice of a for the construction of the hierarchical grid structure. Now consider the hierarchical grid structure for $a = c \cdot U$ for some $c \in [(1/\delta\epsilon)^{1/\epsilon}]$. Rectangle R_i is crossed by a vertical line of level larger than s if and only if it is crossed by a vertical line of level $s+1$. Lines of G_{s+1}^{V} are spaced at distance u_{s+1} from each other, which means that R_i is crossed by a line of G_{s+1}^{V} if and only if the remainder of $c \cdot U$ modulo u_{s+1} is among a set Γ_i of $g_i - 1$ consecutive remainders from $[u_{s+1}]$, being the remainders of the x-coordinates of vertical lines that cross R_i. By (1), Γ_i contains at most $2\epsilon \cdot (1/\delta\epsilon)^{1/\epsilon} + 2 \leq 3\epsilon \cdot (1/\delta\epsilon)^{1/\epsilon}$ multiples of $\sum_{r=1}^{s} u_r$. On the other hand, observe that $0 \leq c \cdot \sum_{r=1}^{s} u_r < u_{s+1}$ for all $c \in [(1/\delta\epsilon)^{1/\epsilon}]$, and that u_r divides u_{r+1} for each r. Hence, $c \cdot U$ gives remainder $c \cdot \sum_{r=1}^{s} u_r$ modulo u_{s+1}, which is always a multiple of $\sum_{r=1}^{s} u_r$. In particular, it follows that the multiples of $\sum_{r=1}^{s} u_r$ contained in Γ_i constitute at most a 3ϵ-fraction of all the remainders modulo u_{s+1} that $c \cdot U$ attains for $c \in [(1/\delta\epsilon)^{1/\epsilon}]$.

Suppose now that $c \in [(1/\delta\epsilon)^{1/\epsilon}]$ is chosen uniformly at random. Let $R_i \in \mathcal{R}_s^{\mathsf{V}} \cap \mathcal{R}_t^{\mathsf{H}}$. By the previous observation, R_i is crossed by a vertical line of level larger than s with probability at most 3ϵ. A similar analysis shows that R_i is crossed by a horizontal line of level larger than t with probability at most 3ϵ. Therefore, R_i is abusive with probability at most 6ϵ, and the total expected weight of the abusive rectangles in $\mathsf{OPT}(\mathcal{R}')$ with respect to $a = c \cdot U$ is bounded by $6\epsilon \cdot \mathsf{OPT}(\mathcal{R}')$. Hence, the value c^*, as claimed in the lemma statement, exists. ◀

We now execute the rest of the algorithm on $\mathcal{R}_c''$ for each $c \in [(1/\delta\epsilon)^{1/\epsilon}]$, and output the best solution obtained overall. This increases the running time of the algorithm by a factor $(1/\delta\epsilon)^{1/\epsilon}$. From Lemma 5, we know that $\mathsf{OPT}(\mathcal{R}_c'') \geq (1 - 6\epsilon)\mathsf{OPT}(\mathcal{R}')$ for $c = c^*$, and thus we lose at most a factor $(1 - 6\epsilon)$. From now on, let $\mathcal{R}'' = \mathcal{R}_c''$ for some $c \in [(1/\delta\epsilon)^{1/\epsilon}]$.

Snapping by shrinking. When considering $\mathcal{R}''$, the lines of G_t^{V} provide a fine division of every rectangle from vertical level t or larger, while no rectangle of smaller vertical level is crossed by them; symmetrically for horizontal grid lines. The idea now is to shrink each rectangle $R_i \in \mathcal{R}''$ so that its vertical sides are aligned with some vertical grid lines of the vertical level of R_i, while the horizontal sides are aligned with some horizontal grid lines of the horizontal level of R_i. This is formalized in the next lemma, which is also similar to Adamaszek et al. [1]. However, in this step there is a subtle but crucial difference. Consider a rectangle $R_i \in \mathcal{R}''$, and suppose $R_i \in \mathcal{R}_s^{\mathsf{V}} \cap \mathcal{R}_t^{\mathsf{H}}$. In [1], R_i is shrunk in the vertical dimension only if $s \geq t$ and in the horizontal dimension only if $t \geq s$. Here we always do both, which will be important for our dynamic programming.

▶ **Lemma 6.** *In polynomial time we can compute a well-behaved family of axis-parallel rectangles $\mathcal{Q}$ that contains one rectangle Q_i for each $R_i \in \mathcal{R}''$, of the same weight w_i as R_i and satisfying the following conditions:*

- *$R_i^{-\delta} \subseteq Q_i \subseteq R_i$ for each $R_i \in \mathcal{R}''$; and*
- *if $R_i \in \mathcal{R}_s^{\mathsf{V}} \cap \mathcal{R}_t^{\mathsf{H}}$, then both vertical sides of Q_i are contained in some vertical grid lines of G_s^{V}, and both horizontal sides of Q_i are contained in some horizontal grid lines of G_t^{H}.*

Proof. Take any $R_i \in \mathcal{R}''$, and suppose $R_i \in \mathcal{R}_s^{\mathsf{V}} \cap \mathcal{R}_t^{\mathsf{H}}$. We define Q_i as the rectangle cut from the plane by the following four lines:

- the left-most and the right-most vertical grid lines of G_s^{V} that cross R_i;
- the bottom-most and the top-most horizontal grid lines of G_t^{H} that cross R_i.

Clearly, we have that $Q_i \subseteq R_i$ and the second condition of the statement is satisfied. We are left with proving that $R_i^{-\delta} \subseteq Q_i$.

Consider first the left side of Q_i, which is contained in the left-most vertical grid line of G_s^{V} that crosses R_i. Since $R_i \in \mathcal{R}_s^{\mathsf{V}}$, we have that $g_i \geq \nu_s$, while the grid lines of G_s^{V} are spaced at distance $u_t = \delta\nu_s/2$ apart. This means that the left-most vertical grid line crossing R_i has the x-coordinate not larger than $x_i^{(1)} + \delta\nu_s/2$, which in turn is not larger than $x_i^{(1)} + \delta g_i/2$. This means that the left side of Q_i is either to the left or at the same x-coordinate as the left side of $R_i^{-\delta}$. An analogous reasoning can be applied to the other three sides of Q_i, thereby proving that $R_i^{-\delta} \subseteq Q_i$.

Note that since $\mathcal{Q}$ is obtained only by shrinking rectangles from $\mathcal{R}''$, it is still the case that no rectangle of $\mathcal{Q}$ is abusive. ◀

By Lemma 4 and Lemma 5, $\mathsf{OPT}(\mathcal{Q}) \geq \mathsf{OPT}(\mathcal{R}'') \geq (1-6\epsilon)\mathsf{OPT}(\mathcal{R}') \geq (1-8\epsilon)\mathsf{OPT}$ if $\mathcal{R}' = \mathcal{R}'_{b^*}$ and $\mathcal{R}'' = \mathcal{R}''_{c^*}$. Hence, the optimum solution for $\mathcal{Q}$ indeed has large enough weight. Moreover, by the first condition of Lemma 6, for any independent set of rectangles in $\mathcal{Q}$, the corresponding rectangles in $\mathcal{R}^{-\delta}$ are also independent. Hence, any solution for MWISR on $\mathcal{Q}$ projects to a solution of the same weight for MWISR with δ-shrinking on $\mathcal{R}$.

From now on we focus on the family $\mathcal{Q}$. For each $t \in \{1, 2, \ldots, p\}$, let $\mathcal{Q}_t^{\mathsf{V}} = \{Q_i \colon R_i \in \mathcal{R}_t^{\mathsf{V}}\}$ and $\mathcal{Q}_t^{\mathsf{H}} = \{Q_i \colon R_i \in \mathcal{R}_t^{\mathsf{H}}\}$.

3.2 Dynamic programming

We now present a dynamic programming algorithm that, given the family $\mathcal{Q}$ constructed in the previous section, computes the value $\mathsf{OPT}(\mathcal{Q})$. An optimum solution can be recovered from the run of the dynamic program using standard methods, and hence for simplicity we omit this aspect in the description.

We describe the algorithm as *backtracking with memoization*. That is, subproblems are solved by recursion, but once a subproblem has been solved once, the optimum value for it is stored in a map (is *memoized*), and further calls to solving this subproblem will only retrieve the memoized optimum value, rather than solve the subproblem again. Solving each subproblem (excluding recursive subcalls) takes time $f(\delta, \epsilon) \cdot n^{\mathcal{O}(1)}$ for some computable function f, and we argue that at most $g(\delta, \epsilon) \cdot (nN)^{\mathcal{O}(1)}$ subproblems are solved in total, for some other computable function g. This ensures the promised running time of the algorithm.

We first define subproblems. A *subproblem* is a tuple $I = (s, t, x_1, x_2, y_1, y_2)$, where the meaning of the entries is as follows. The pair $(s, t) \in \{1, \ldots, p\} \times \{1, \ldots, p\}$ is the *level* of the subproblem, which consists of the vertical level s and the horizontal level t. The numbers x_1, x_2, y_1, y_2 are integers satisfying $x_1 < x_2 \leq x_1 + (1/\delta\epsilon)^{1/\epsilon}$ and $y_1 < y_2 \leq y_1 + (1/\delta\epsilon)^{1/\epsilon}$. Integers x_1, x_2 are the lower and upper *vertical offsets*, respectively, while y_1, y_2 are the

lower and upper *horizontal offsets*, respectively. The *area covered* by subproblem $I = (s, t, x_1, x_2, y_1, y_2)$ is the rectangle

$$A_I = (a + x_1 \cdot u_s, a + x_2 \cdot u_s) \times (a + y_1 \cdot u_t, a + y_2 \cdot u_t).$$

In other words, (x_1, x_2, y_1, y_2) define the offsets of the four grid lines – two from G_s^{V} and two from G_t^{H} – that cut out A_I from the plane.

For a subproblem I, let $\mathcal{Q}_I$ be the family of all rectangles from $\mathcal{Q}$ that are contained in A_I. The next check follows from a simple calculation of parameters.

▶ **Lemma 7 (♠).** *If subproblem I has level (s, t), then $\mathcal{Q}_I \subseteq \bigcup_{s' \leq s,\, t' \leq t} \mathcal{Q}_{s'}^{\mathsf{V}} \cap \mathcal{Q}_{t'}^{\mathsf{H}}$.*

For a subproblem I, we define the *value* of I, denoted $\mathsf{Value}(I)$, as the maximum weight of a subfamily of $\mathcal{Q}_I$ that is independent. We show that there is a subproblem that encompasses the whole instance. Then, we show how to *solve* each subproblem I, that is, to compute $\mathsf{Value}(I)$, using recursion.

▶ **Lemma 8 (♠).** *There is a subproblem I_{all} of level (p, p), computable in constant time, such that $A_{I_{\mathsf{all}}} \supseteq (1, L) \times (1, L)$. Consequently, $\mathsf{OPT}(\mathcal{Q}) = \mathsf{Value}(I_{\mathsf{all}})$.*

Next comes the crucial point: we show how to *solve* each subproblem I, that is, to compute $\mathsf{Value}(I)$, using recursion.

▶ **Lemma 9 (♠).** *A subproblem I of level (s, t) can be solved using $f(\delta, \epsilon)$ calls to solving subproblems of levels $(s-1, t)$, $(s, t-1)$, and $(s-1, t-1)$, for some computable function f. Moreover, the time needed for this computation, excluding the time spent in the recursive calls, is at most $f(\delta, \epsilon) \cdot n$.*

Proof sketch. Consider all vertical lines of level s and all horizontal lines of level t that cross the rectangle A_I; their number is bounded by $(1/\delta\epsilon)^{1/\epsilon}$. These lines partition A_I into at most $(1/\delta\epsilon)^{2/\epsilon}$ smaller *cells* in a natural manner. Consider an independent set $\mathcal{S} \subseteq \mathcal{Q}_I$ such that $w(\mathcal{S}) = \mathsf{Value}(I)$. By obtaining a well-behaved family and applying the snapping procedure, we have the following structure; see Figure 2. Each rectangle from $\mathcal{S}$ of level (s, t) just occupies a rectangle of cells, each of them entirely. Whenever a rectangle from $\mathcal{S}$ is of level (s', t) for some $s' < s$, it is contained in a single column of cells, and its horizontal sides are aligned with some horizontal lines of the grid of cells. A symmetrical claim holds for rectangles from $\mathcal{S}$ of level (s, t') for any $t' < t$. Finally, any rectangle of $\mathcal{S}$ of level (s', t') for $s' < s$ and $t' < t$ is contained in a single cell.

It can be then easily seen that the whole grid of cells admits a partition into "boxes" such that each rectangle of $\mathcal{S}$ fits into a single box. Each box that is not contained in one column or in one row must be filled with a single rectangle of level (s, t), and we can greedily take the heaviest such rectangle. Each other box defines a subproblem of level (s', t') where $s' \leq s$, $t' \leq t$, and one of these inequalities is strict. Hence, an optimum solution for such a box can be computed using a recursive call. Therefore, the algorithm enumerates all partitions of the cells into boxes, and for each of them computes a candidate value using recursive subcalls; the value of I is the largest among the candidates. ◀

Finally, to bound the running time of the algorithm, we prove that there is only a small number of subproblems I for which $\mathcal{Q}_I$ is nonempty. Obviously, only such subproblems are necessary to solve, as the others have value 0.

▶ **Lemma 10 (♠).** *The number of subproblems I for which $\mathcal{Q}_I$ is nonempty is at most $81 \cdot (1/\delta\epsilon)^{4/\epsilon} \cdot |\mathcal{Q}| p^2$.*

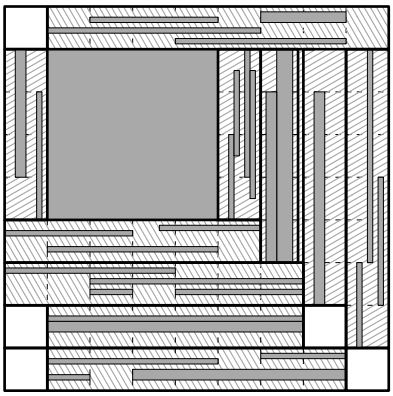

Figure 2 The partition of A_I into large, horizontal, vertical, and small cells. The figure is a slightly modified figure from [1].

Having gathered all the tools, we can describe the algorithm. To compute $\mathsf{OPT}(\mathcal{Q})$, it is sufficient to compute $\mathsf{Value}(I_{\mathsf{all}})$ for the subproblem I_{all} given by Lemma 8. For this we use backtracking with memoization. Starting from I_{all}, we recursively solve subproblems as explained in Lemma 9. Whenever $\mathsf{Value}(I)$ has been computed for some subproblem I, then this value is memoized in a map, and further calls to solving I will only return the value retrieved from the map, instead of recomputing the value again. Furthermore, whenever we attempt to compute $\mathsf{Value}(I)$ for a subproblem I for which $\mathcal{Q}_I$ is empty, we immediately return 0 instead of applying the procedure of Lemma 9. Therefore, the total running time of the algorithm is upper bounded by the number of subproblems I for which $\mathcal{Q}_I$ is nonempty, times the time spent on internal computations for each of them, including checking whether the respective family $\mathcal{Q}_I$ is empty and whether $\mathsf{Value}(I)$ has already been memoized. The first factor is bounded by $81 \cdot (1/\delta\epsilon)^{4/\epsilon} \cdot |\mathcal{Q}|p^2 \leq 81 \cdot (1/\delta\epsilon)^{4/\epsilon} \cdot nN^2$ due to Lemma 10. The second factor is bounded by $f(\delta,\epsilon) \cdot n^d$ for some constant d, due to Lemma 9. Hence, the running time of the whole algorithm is $f(\delta,\epsilon) \cdot (nN)^c$ for some computable function f and constant c. As mentioned before, the algorithm can be trivially adjusted to also compute an independent set of weight $\mathsf{Value}(I_{\mathsf{all}})$ by storing the value of each subproblem together with some independent set that certifies this value.

Summarizing, the dynamic programming described above computes an independent set in $\mathcal{Q}$ of weight $\mathsf{OPT}(\mathcal{Q})$ in time $f(\delta,\epsilon) \cdot (nN)^c$. As argued in the previous section, such an independent set projects to an independent set of the same weight in $\mathcal{R}^{-\delta}$, and $\mathsf{OPT}(\mathcal{Q}) \geq (1-8\epsilon)\mathsf{OPT}$ holds. This concludes the proof of Theorem 2.

4 Kernelization results

In this section we discuss kernelization for the case when the input family consists of squares. We first clarify the definition of a kernel, and then present the results.

Definition of kernel. The classic definition of kernelization is tailored to decision problems. Extending it to optimization problems in a weighted setting is often problematic, and making it compatible with the δ-shrinking model complicate it even more. Hence, we explicitly define kernelization for MWISR in the shrinking model, bearing in mind the main principle of kernelization: solving the kernel should project to a solution for the original instance.

▶ **Definition 11.** Let k be a non-negative integer, let $\delta \in (0,1)$, and let $\mathcal{R}$ be a family of axis-parallel rectangles. Then, a *kernel for* $(\mathcal{R}, k, \delta)$ is a polynomial-time computable subfamily $\mathcal{Q} \subseteq \mathcal{R}$ such that $|\mathcal{Q}| \leq f(k,\delta)$ for a computable function f, called the *size* of the kernel, and:

$$\mathsf{OPT}_k(\mathcal{Q}^{-\delta}) \geq \mathsf{OPT}_k(\mathcal{R}).$$

Thus, MWISR on $\mathcal{R}$ with the δ-shrinking relaxation may be solved by solving MWISR on $\mathcal{Q}^{-\delta}$ (without the δ-shrinking relaxation). If one wishes to use an algorithm for the shrinking relaxation on the kernel, then applying it to $\mathcal{Q}^{-\delta}$ with parameter δ will yield a subfamily $\mathcal{S}$ of size k such that $\mathcal{S}^{-2\delta}$ is independent and $w(\mathcal{S}) \geq \mathsf{OPT}_k(\mathcal{Q}^{-\delta}) \geq \mathsf{OPT}_k(\mathcal{R})$. Hence, this solves the original problem for 2δ-shrinking and we can rescale δ accordingly.

Definition 11 lacks one aspect of the classic notion of kernelization. Namely, the weights and the coordinates of the rectangles in the kernel are inherited from the original instance, so their bit encoding may not be bounded in terms of k and δ. We prefer to work with Definition 11, because it focuses our efforts on the core combinatorial aspects of our kernelization procedures. However, in the full version we argue that the sizes of the bit encodings of both the weights and the coordinates essentially can be reduced to polynomials in k and $1/\delta$ [14].

Results. The following theorem summarizes our kernelization results.

▶ **Theorem 12 (♠).** *Given a non-negative integer k, $\delta \in (0,1)$, and a family of axis-parallel squares $\mathcal{R}$, the following kernels for $(\mathcal{R}, k, \delta)$ can be computed in polynomial time:*

1. *If $\mathcal{R}$ consists of unit squares of uniform weight, then there is a kernel of size $\leq 16k/\delta^2$.*
2. *If $\mathcal{R}$ consists of unit squares of non-uniform weight, then there is a kernel of size $\leq 64k/\delta^2$.*
3. *If $\mathcal{R}$ consists of squares of non-uniform size, but of uniform weight, then there is a kernel of size $\mathcal{O}(k^2 \cdot \frac{\log(1/\delta)}{\delta^3})$.*

We now briefly sketch the main ideas. Consider first the simplest case of unit squares of uniform weight. Suppose two squares R_i and R_j are very close to each other: their centers are at distance less than $\delta/2$ in the ℓ_∞-metric (we will say that R_i and R_j are $(\delta/2)$-*close*). Then it is not hard to prove that $R_j^{-\delta} \subseteq R_i$, so intuitively, in the δ-shrinking model R_j is always a valid substitute for R_i. This allows for the following greedy strategy. Compute an inclusion-wise maximal subfamily $\mathcal{Q} \subseteq \mathcal{R}$ such that the centers of squares in $\mathcal{Q}$ are pairwise at distance at least $\delta/2$. By maximality, for every square $R_i \in \mathcal{R}$, there is a square $\phi(R_i) \in \mathcal{Q}$ that is $(\delta/2)$-close to R_i. By the observation above, the mapping ϕ maps every independent set in $\mathcal{R}$ to a subset of $\mathcal{Q}$ of the same size that is independent after δ-shrinking. Consequently, $\mathsf{OPT}_k(\mathcal{Q}^{-\delta}) \geq \mathsf{OPT}_k(\mathcal{R})$ and we may work with $\mathcal{Q}$ instead. However, the fact that squares of $\mathcal{Q}$ have centers pairwise far from each other immediately shows that the intersection graph of $\mathcal{Q}^{-\delta}$ has maximum degree bounded by a function of $1/\delta$ (similar to [4]). Then it is a standard exercise to give a linear kernel for Independent Set.

For unit squares of non-uniform weight, we follow the same strategy, but we construct $\mathcal{Q}$ greedily by iteratively taking the heaviest square and removing all squares that are $(\delta/2)$-close to it. This ensures that the substitute $\phi(R_i)$ is always at least as heavy as R_i. The maximum degree of the intersection graph of $\mathcal{Q}^{-\delta}$ can be bounded in the same manner. There is also a linear kernel for Maximum Weight Independent Set on bounded degree graphs, following the observation: Iterate k times the procedure of picking the heaviest vertex and removing all vertices at distance at most 2 from it. Then there is some maximum-weight independent set of size at most k that is contained in the removed vertices.

Finally, in the case of squares of non-uniform size and uniform weight, the following observation is crucial: if there are two squares R_i and R_j whose sizes differ by a multiplicative factor at least $2/\delta$, then either one is contained in the other, or they become disjoint after δ-shrinking. Observe that we may assume that the input does not contain any pair of squares where one is contained in the other, since the smaller one can be always selected instead of the larger. Hence, squares of very different sizes become disjoint after δ-shrinking. We partition the squares into levels according to the magnitude of their side lengths, and apply the following win-win approach. If many levels are non-empty, then we can find k of them so that picking one square from each yields an independent set of size k after δ-shrinking. Otherwise, only few levels are non-empty, and we can treat each level separately using essentially the same methods as for unit squares.

We conclude by discussing some applications. By composing our kernelization algorithms with the parameterized algorithm of Marx and Pilipczuk [13] for finding the heaviest k-independent set of polygons in the plane, we obtain algorithms with running time $(k/\delta)^{\mathcal{O}(\sqrt{k})} \cdot (nN)^{\mathcal{O}(1)}$ for all the problems encompassed by Theorem 12; this is much faster than the algorithm of Theorem 1. Also, some of our intermediate tools can be combined with the results of Alber and Fiala [4], yielding algorithms with running time $2^{\mathcal{O}_\delta(\sqrt{k})} \cdot (nN)^{\mathcal{O}(1)}$ for unit squares of non-uniform weight. We also obtain a faster EPTAS than Theorem 2 for squares of uniform size or uniform weight, for which the exponent depends only linearly on $1/\epsilon$. A precise description of these results can be found in the full version [14].

5 Conclusions

In this paper we have initiated the study of the shrinking model for parameterized geometric Independent Set, by giving FPT algorithms and polynomial kernels for the most basic variants. Most importantly, we have showcased that the shrinking model leads to robust tractability of problems that without this relaxation are hard from the parameterized perspective. We hope that this is the start of an interesting research direction, as our work raises several concrete open problems. Can our FPT algorithm and EPTAS for axis-parallel rectangles (Theorems 2 and 1) be generalized to arbitrary convex polygons, as is the case for the PTAS [15]? Recall that it was important for our algorithm that via shrinking we can align all edges of each rectangle with grid lines of suitable granularity. Then we could partition the plane recursively along these grid lines. Such an alignment is no longer possible for polygons, not even if each polygon is essentially a diagonal straight line segment. Instead, the PTAS in [15] crucially relies on guessing separators described via $\mathcal{O}_\epsilon(1)$ input polygons (and additionally $\mathcal{O}_\epsilon(1)$ grid lines). The total number of candidates for such separators is $n^{\mathcal{O}_\epsilon(1)}$, which leads to the running time of $n^{\mathcal{O}_\epsilon(1)}$ of the algorithm. Further, is it possible to improve the running time of our FPT algorithm to subexponential, that is, $2^{o(k)} \cdot (nN)^{\mathcal{O}(1)}$ for every fixed δ? What about polynomial kernels, i.e., kernelization procedures with polynomial output guarantees, for more complex objects than squares? Here, it seems that our arguments apply mutatis mutandis to e.g. (unit) disks instead of (unit) squares, but it is conceivable that even the general setting of convex polygons can be treated, for an appropriate definition of shrinking. Also, is there a polynomial kernel for squares of non-uniform size and non-uniform weight? This is not addressed in its full generality by our kernelization algorithms. Finally, can one give limits to tractability in the shrinking model, by showing W[1]-hardness or the nonexistence of polynomial kernels?

Acknowledgements. The first author thanks Dániel Marx for discussions about the setting.

References

1. Anna Adamaszek, Parinya Chalermsook, and Andreas Wiese. How to Tame Rectangles: Solving Independent Set and Coloring of Rectangles via Shrinking. In *Proc. APPROX/RANDOM 2015*, volume 40 of *LIPIcs*, pages 43–60. Schloss Dagstuhl, 2015. doi:10.4230/LIPIcs.APPROX-RANDOM.2015.43.
2. Anna Adamaszek and Andreas Wiese. Approximation schemes for maximum weight independent set of rectangles. In *Proc. FOCS 2013*, pages 400–409. IEEE, 2013. doi:10.1109/FOCS.2013.50.
3. Anna Adamaszek and Andreas Wiese. A QPTAS for maximum weight independent set of polygons with polylogarithmically many vertices. In *Proc. SODA 2014*, pages 645–656. SIAM, 2014. doi:10.1137/1.9781611973402.49.
4. Jochen Alber and Jiří Fiala. Geometric separation and exact solutions for the parameterized independent set problem on disk graphs. *J. Algorithms*, 52(2):134–151, 2004. doi:10.1016/j.jalgor.2003.10.001.
5. Parinya Chalermsook and Julia Chuzhoy. Maximum independent set of rectangles. In *Proc. SODA 2009*, pages 892–901. SIAM, 2009.
6. Timothy M. Chan. Polynomial-time approximation schemes for packing and piercing fat objects. *Journal of Algorithms*, 46(2):178–189, 2003. doi:10.1016/S0196-6774(02)00294-8.
7. Timothy M. Chan and Sariel Har-Peled. Approximation algorithms for maximum independent set of pseudo-disks. *Discrete & Comp. Geometry*, 48(2):373–392, 2012. doi:10.1007/s00454-012-9417-5.
8. Julia Chuzhoy and Alina Ene. On approximating maximum independent set of rectangles. In *Proc. FOCS 2016*, pages 820–829. IEEE, 2016. doi:10.1109/FOCS.2016.92.
9. Rod G. Downey and Michael R. Fellows. Fixed-parameter tractability and completeness II: On completeness for W[1]. *Theoretical Computer Science*, 141(1):109–131, 1995. doi:10.1016/0304-3975(94)00097-3.
10. Thomas Erlebach, Klaus Jansen, and Eike Seidel. Polynomial-time approximation schemes for geometric intersection graphs. *SIAM J. on Computing*, 34(6):1302–1323, 2005. doi:10.1137/S0097539702402676.
11. Sariel Har-Peled. Quasi-polynomial time approximation scheme for sparse subsets of polygons. In *Proc. SOCG 2014*, pages 120–129. ACM, 2014. doi:10.1145/2582112.2582157.
12. Dániel Marx. Efficient approximation schemes for geometric problems? In *Proc. ESA 2005*, volume 3669 of *LNCS*, pages 448–459. Springer, 2005. doi:10.1007/11561071_41.
13. Dániel Marx and Michał Pilipczuk. Optimal parameterized algorithms for planar facility location problems using Voronoi diagrams. In *Proc. ESA 2015*, volume 9294 of *LNCS*, pages 865–877. Springer, 2015. doi:10.1007/978-3-662-48350-3_72.
14. Michał Pilipczuk, Erik Jan van Leeuwen, and Andreas Wiese. Approximation and parameterized algorithms for geometric independent set with shrinking. *CoRR*, abs/1611.06501, 2016. arXiv:1611.06501.
15. Andreas Wiese. Independent set of convex polygons: From n^ϵ to $1+\epsilon$ via shrinking. In *Proc. LATIN 2016*, volume 9644 of *LNCS*, pages 700–711. Springer, 2016. doi:10.1007/978-3-662-49529-2_52.
16. David Zuckerman. Linear degree extractors and the inapproximability of max clique and chromatic number. *Theory of Computing*, 3:103–128, 2007. doi:10.4086/toc.2007.v003a006.