2016 IEEE 57th Annual Symposium on Foundations of Computer Science

On Approximating Maximum Independent Set of Rectangles

Julia Chuzhoy
Toyota Technological Institute at Chicago
Chicago, IL, USA
Email: cjulia@ttic.edu

Alina Ene
Dept. of Computer Science, Boston University
Boston, MA, USA
Email: aene@bu.edu

***Abstract*—We study the Maximum Independent Set of Rectangles (MISR) problem: given a set of n axis-parallel rectangles, find a largest-cardinality subset of the rectangles, such that no two of them overlap. MISR is a basic geometric optimization problem with many applications, that has been studied extensively. Until recently, the best approximation algorithm for it achieved an $O(\log \log n)$-approximation factor. In a recent breakthrough, Adamaszek and Wiese provided a *quasi-polynomial* time approximation scheme: a $(1-\epsilon)$-approximation algorithm with running time $n^{O(\text{poly}(\log n)/\epsilon)}$. Despite this result, obtaining a PTAS or even a polynomial-time constant-factor approximation remains a challenging open problem. In this paper we make progress towards this goal by providing an algorithm for MISR that achieves a $(1-\epsilon)$-approximation in time $n^{O(\text{poly}(\log \log n/\epsilon))}$. We introduce several new technical ideas, that we hope will lead to further progress on this and related problems.**

I. Introduction

In the Maximum Independent Set of Rectangles (MISR) problem, the input is a set $\mathcal{R}$ of n axis-parallel rectangles, and the goal is to find a maximum-cardinality subset of the rectangles, such that no two of them overlap. MISR is a fundamental geometric optimization problem with applications to map labeling [3], [10], resource allocation [20], and data mining [18], [14], [19]. It is also a special case of the classical Maximum Independent Set problem, where the input is an n-vertex graph G, and the goal is to find a maximum-cardinality subset S of its vertices, so that no edge of G has both endpoints in S. Maximum Independent Set is one of the most fundamental and extensively studied problems in combinatorial optimization. Unfortunately, it is known to be very difficult to approximate: the problem does not have an $n^{1-\epsilon}$-approximation algorithm for any constant ϵ unless NP $=$ ZPP [16], and the best current positive result gives an $O(n/\log^2 n)$-approximation algorithm [5]. It is therefore natural to focus on important classes of special cases of the problem, where better approximation guarantees may be achievable. This direction has proved to be especially fruitful for instances stemming from geometric objects in the plane. Results in this area range from Polynomial-Time Approximation Schemes (PTAS) for "fat objects", such as disks and squares [11], to an n^{ϵ}-approximation for arbitrary geometric shapes [13]. Unfortunately, the techniques used in algorithms for fat geometric objects seem to break down for other geometric shapes. Rectangles are among the simplest shapes that are not fat, which puts MISR close to the boundary of the class of geometric problems for which PTAS is achievable with current techniques.

MISR is a basic geometric variant of Independent Set, and rectangles seem to be among the simplest shapes that capture several of the key difficulties associated with objects that are not fat. It is then not surprising that MISR has attracted a considerable amount of interest from various research communities. Since the problem is known to be NP-hard [12], [17], the main focus has been on designing approximation algorithms. Several groups of researches have independently suggested $O(\log n)$-approximation algorithms for MISR [3], [18], [22], and Berman et al. [4] showed that there is a $\lceil \log_k n \rceil$ approximation for any fixed k. More recently an $O(\log \log n)$-approximation was shown [7], that remains the best current approximation algorithm that runs in polynomial time. The result of [7] also gives a matching upper bound on the integrality gap of the natural LP relaxation for MISR. The best current lower bound on the integrality gap of this LP relaxation is a small constant, and understanding this gap is a long-standing open question with a beautiful connection to rectangle coloring; see [6] and references therein. In a recent breakthrough, Adamaszek and Wiese [1] designed a Quasi-Polynomial Time Approximation Scheme (QPTAS) for MISR: namely, a $(1-\epsilon)^1$-approximation algorithm with running time $n^{O(\text{poly}(\log n/\epsilon))}$, using completely different techniques. Their result can be seen as a significant evidence that MISR may admit a PTAS. However, obtaining a PTAS, or even an efficient constant-factor approximation remains elusive for now.

In this paper, we make progress towards this goal, by providing an algorithm for MISR that achieves a $(1-\epsilon)$-approximation and runs in time $n^{O\left((\log \log n/\epsilon)^4\right)}$. We introduce several new technical ideas that we hope will lead to further progress on this and related problems.

The MISR problem seems to be central to understanding

[1]So far we have followed the convention that approximation factors of algorithms are greater than 1, but for QPTAS-type results it is more convenient for us to switch to approximation factors of the form $(1-\epsilon)$.

0272-5428/16 $31.00 © 2016 IEEE
DOI 10.1109/FOCS.2016.92

several other geometric problems. The work of [1] has been very influential, and has lead to several new results, including, for example, a QPTAS for Maximum Independent Set of Polygons [2], [15], and QPTAS for several geometric Set Cover problems [21].

Other related work. Several important special cases of MISR have been studied extensively. In particular, there is a PTAS for squares — and more generally, rectangles with bounded aspect ratio [11] — and large rectangles whose width or height is within a constant factor of the size of the bounding box that encloses the entire input [1]. We note that a more general weighted version of the MISR problem has also been considered, where all input rectangles are associated with non-negative weights, and the goal is to find a maximum-weight subset of non-overlapping rectangles. As mentioned earlier, there are several algorithms for MISR that achieve an $O(\log n)$-approximation, and these results hold in the weighted setting as well. The long-standing $O(\log n)$-approximation was improved in the work of Chan and Har-Peled that achieved an $O(\log n/\log\log n)$-approximation for the weighted problem [8]. This result remains the best polynomial-time approximation algorithm for the weighted problem, as the $O(\log\log n)$-approximation algorithm of [7] only applies to the unweighted version of MISR. The work of Adamaszek and Wiese [1] extends to the weighted version and provides a QPTAS for it as well. There seem to be several technical difficulties in extending our results to the weighted version of the problem, and we leave this as an open problem.

Our Results and Techniques. Our main result is summarized in the following theorem.

Theorem I.1 *There is an algorithm for the* MISR *problem that, given any set $\mathcal{R}$ of n axis-parallel rectangles and a parameter $0 < \epsilon < 1$, computes a $(1-\epsilon)$-approximate solution to instance $\mathcal{R}$, in time $n^{O((\log\log n/\epsilon)^4)}$.*

In order to put our techniques in context, we first give a high-level overview of the approach of Adamaszek and Wiese [1]. The description here is somewhat over-simplified, and is different from the description of [1], though the algorithm is essentially the same. Their approach is based on dynamic programming, and uses the divide-and-conquer paradigm. Starting with the initial set of rectangles, the algorithm recursively partitions the input into smaller sub-instances. A key insight is the use of closed polygonal curves to partition the instances: given such a curve, one can discard the rectangles that intersect the curve; the remaining rectangles can be naturally partitioned into two sub-instances, one containing the rectangles lying in the interior of the curve and the other containing rectangles lying outside the curve[2]. Adamaszek and Wiese show that for every set $\mathcal{R}^*$ of non-overlapping rectangles and an integral parameter L, there is a closed polygonal curve C, whose edges are parallel to the axes, so that C has at most L corners; the number of rectangles of $\mathcal{R}^*$ intersecting C is at most $O(|\mathcal{R}^*|/L)$; and at most a $3/4$-fraction of the rectangles of $\mathcal{R}^*$ lie on either side of the curve C. We call such a curve C a *balanced L-corner partitioning curve* for set $\mathcal{R}^*$. Given any subset $\mathcal{R}' \subseteq \mathcal{R}$ of rectangles, we denote by $\mathsf{OPT}(\mathcal{R}')$ the optimal solution to instance $\mathcal{R}'$, and we denote $\mathsf{OPT} = \mathsf{OPT}(\mathcal{R})$. Throughout this exposition, all polygons and polygonal curves have all their edges parallel to the axes. We sometimes refer to the number of corners of a polygon as its boundary complexity.

The approach of [1] can now be described as follows. Let $L = \Theta(\log n/\epsilon)$ and $L^* = \Theta(L \cdot \log n)$. The algorithm uses dynamic programming. Every entry of the dynamic programming table T corresponds to a polygon P that has at most L^* corners. The entry $T[P]$ will contain an approximate solution to instance $\mathcal{R}(P)$, that consists of all rectangles $R \in \mathcal{R}$ with $R \subseteq P$. We say that P defines a basic instance if $|\mathsf{OPT}(\mathcal{R}(P))| \leq \log n$. We can check whether P defines a basic instance, and if so, find an optimal solution for it in time $n^{O(\log n)}$ via exhaustive search. In order to compute the entry $T[P]$ where $\mathcal{R}(P)$ is a non-basic instance, we go over all pairs $P', P'' \subsetneq P$ of polygons with $P' \cap P'' = \emptyset$, such that P' and P'' have at most L^* corners each, and we let $T[P]$ contain the best solution $T[P'] \cup T[P'']$ among all such possible pairs of polygons. In order to analyze the approximation factor achieved by this algorithm, we build a partitioning tree, that will simulate an idealized run of the dynamic program. Every vertex v of the partitioning tree is associated with some polygon $P(v)$ that has at most L^* corners, and stores some solution to instance $\mathcal{R}(v) = \mathcal{R}(P(v))$, consisting of all rectangles $R \in \mathcal{R}$ with $R \subseteq P(v)$. For the root vertex of the tree, the corresponding polygon P is the bounding box of our instance. Given any leaf vertex v of the current tree, such that the instance $\mathcal{R}(v)$ is non-basic, we add two children v', v'' to v, whose corresponding polygons P' and P'' are obtained by partitioning P with the balanced L-corner partitioning curve C for the set $\mathsf{OPT}(\mathcal{R}(v))$ of rectangles. We terminate the algorithm when for every leaf vertex v, $\mathcal{R}(v)$ is a basic instance. It is easy to verify that the height of the resulting tree is $O(\log n)$. The polygon associated with the root vertex of the tree has 4 corners, and for every $1 \leq i \leq \log n$, the polygons associated with the vertices lying at distance exactly i from the root have at most $4 + iL$ corners. Therefore, every polygon associated with the vertices of the tree has at most L^* corners, and

[2]The sub-instances could have holes. In order to simplify the exposition, instead of working with instances with holes, we later introduce what we call "fake rectangles" and use them to partition the instance.

corresponds to some entry of the dynamic programming table. Once the tree is constructed, we compute solutions to sub-instances associated with its vertices, as follows. For every leaf v of the tree, the solution associated with v is the optimal solution to instance $\mathcal{R}(v)$; for an inner vertex v of the tree with children v' and v'', the solution associated with v is the union of the solutions associated with v' and v''. Let $\mathcal{R}'$ be the solution to the MISR problem associated with the root vertex of the tree. Then it is easy to see that the solution computed by the dynamic programming algorithm has value at least $|\mathcal{R}'|$. Moreover, from our choice of parameters, $|\mathcal{R}'| \geq |\mathsf{OPT}(\mathcal{R})|(1-\epsilon)$. This is since for every inner vertex v of the tree, with children v' and v'', the loss incurred by the partitioning procedure at v, $\lambda(v) = |\mathsf{OPT}(\mathcal{R}(v))| - |\mathsf{OPT}(\mathcal{R}(v'))| - |\mathsf{OPT}(\mathcal{R}(v''))| \leq |\mathsf{OPT}(\mathcal{R}(v))|/L \leq \epsilon|\mathsf{OPT}(\mathcal{R}(v))|/\log n$. It is then easy to verify that the total loss of all vertices that lie within distance exactly i from the root, for any fixed $0 \leq i \leq \log n$, is at most $\epsilon|\mathsf{OPT}(\mathcal{R})|/\log n$, and the total loss of all vertices is at most $\epsilon|\mathsf{OPT}(\mathcal{R})|$. It is also immediate to verify that the value of the solution stored at the root vertex of the tree is at least $|\mathsf{OPT}(\mathcal{R})| - \sum_v \lambda(v)$, and so we obtain a $(1-\epsilon)$-approximation.

In order to bound the running time of the algorithm, it is not hard to show by a standard transformation to the problem input, that it is enough to consider polygons P whose corners have integral coordinates between 1 and $2n$. The number of entries of the dynamic programming table, and the running time of the algorithm, are then bounded by $n^{O(L^*)} = n^{O(\log^2 n/\epsilon)}$. As a warmup, we show that this running time can be improved to $n^{O(\log n/\epsilon^3)}$. The idea is that, instead of computing a balanced L-corner partition of the set $\mathsf{OPT}(\mathcal{R}(P))$ of rectangles, we can compute a different partition that reduces the boundary complexities of the two resulting polygons. If P has boundary complexity greater than L, then we can compute a polygonal curve C, that partitions P into polygons P' and P'', such that the number of corners of each of the two polygons P' and P'' is smaller than the number of corners of P by a constant factor, and $|\mathsf{OPT}(\mathcal{R}(P))| - |\mathsf{OPT}(\mathcal{R}(P'))| - |\mathsf{OPT}(\mathcal{R}(P''))| \leq f(L) \cdot |\mathsf{OPT}(\mathcal{R}(P))|$, where $f(L) = O(1/L)$. In our partitioning tree we can then alternate between computing balanced L-corner curves, and computing partitions that reduce the boundary complexities of the polygons, so that the number of corners of the polygons does not accumulate as we go down the tree. This allows us to set $L^* = L = \Theta(\log n/\epsilon)$, and obtain a running time of $n^{O(\log n/\epsilon^3)}$.

The bottleneck in the running time of the above algorithm is the number of entries in the dynamic programming table, which is $n^{O(L^*)}$, where L^* is the number of corners that we allow for our polygons, and the term n appears since there are $\Theta(n^2)$ choices for each such corner. In order to improve the running time, it is natural to try one of the following two approaches: either (i) decrease the parameter L^*, or (ii) restrict the number of options for choosing each corner. The latter approach can be, for example, implemented by discretization: we can construct an $(N \times N)$-grid G, where N is small enough. We say that a polygon P is *aligned with* G if all corners of P are also vertices of G. We can then restrict the polygons we consider to the ones that are aligned with G. Unfortunately, neither of these approaches works directly. For the first approach, since the depth of the partitioning tree is $\Theta(\log n)$, we can only afford to lose an $O(\epsilon/\log n)$-fraction of rectangles from the optimal solution in every iteration, that is, on average, for an inner vertex v of the tree $\lambda(v) \leq O(\epsilon/\log n) \cdot |\mathsf{OPT}(\mathcal{R}(v))|$ must hold. It is not hard to show that this constraint forces us to allow the partitioning curve C to have as many as $\Omega(\log n/\epsilon)$ corners, and so in general $L^* = \Omega(\log n/\epsilon)$ must hold. For the second approach, over the course of our algorithm, we will need to handle sub-instances whose optimal solution values are small relatively to $|\mathsf{OPT}|$, and their corresponding polygons have small areas. If we construct an $(N \times N)$-grid G, with $N << n$, then polygons that are aligned with G cannot capture all such instances.

In order to better illustrate our approach, we start by showing a $(1-\epsilon)$-approximation algorithm with running time $n^{O(\sqrt{\log n}/\epsilon^3)}$. This algorithm already needs to overcome the barriers described above, and will motivate our final algorithm. Consider the divide-and-conquer view of the algorithm, like the one we described in the construction of the partitioning tree. We can partition this algorithm into $\Theta(\sqrt{\log n})$ phases, where the values of the optimal solutions $|\mathsf{OPT}(\mathcal{R}(v))|$ of instances $\mathcal{R}(v)$ considered in every phase go down by a factor of approximately $2^{\sqrt{\log n}}$. In other words, if we consider the partitioning tree, and we call all vertices at distance exactly i from the root of the tree *level-i vertices*, then every phase of the algorithm roughly contains $\Theta(\sqrt{\log n})$ consecutive levels of the tree. Therefore, the number of such phases is only $O(\sqrt{\log n})$, and so we can afford to lose an $\Theta(\epsilon/\sqrt{\log n})$-fraction of the rectangles from the optimal solution in every phase. At the end of every phase, for every polygon P defining one of the resulting instances $\mathcal{R}(P)$ of the problem, we can then afford to repeatedly partition P into sub-polygons, reducing their boundary complexity to $O(\sqrt{\log n}/\epsilon)$. This allows us to use polygons with only $L_1 = \Theta(\sqrt{\log n}/\epsilon)$ corners as the "interface" between the different phases. Within each phase, we still need to allow the polygons we consider to have $L_2 = \Theta(\log n/\epsilon)$ corners. However, now we can exploit the second approach: since the values of the optimal solutions of all instances considered within a single phase are relatively close to each other - within a factor of $2^{\Theta(\sqrt{\log n})}$, we can employ discretization, by constructing a grid with $2^{O(\sqrt{\log n})}$ vertical and horizontal lines, and requiring that polygons considered in the phase are aligned with this grid.

To summarize, we use a two-level recursive construction. The set of level-1 polygons (that intuitively serve as the interface between the phases), contains all polygons whose corners have integral coordinates between 1 and $2n$, and they have at most $L_1 = \Theta(\sqrt{\log n}/\epsilon)$ corners. The number of such polygons is $n^{O(L_1)} = n^{O(\sqrt{\log n})/\epsilon}$. Given a level-1 polygon P, we construct a collection $\mathcal{C}(P)$ of level-2 polygons $P' \subseteq P$. We start by constructing a grid G_P that discretizes P, and has $2^{O(\sqrt{\log n})}$ vertical and horizontal lines. The grid has the property that for every vertical strip S of the grid, the value of the optimal solution of the instance $\mathcal{R}(P \cap S)$ is bounded by $|\mathsf{OPT}(\mathcal{R}(P))|/2^{\Theta(\sqrt{\log n})}$, and the same holds for the horizontal strips. Set $\mathcal{C}(P)$ contains all polygons $P' \subseteq P$ that have at most $L_2 = \Theta(\log n/\epsilon)$ corners, so that P' is aligned with G_P. The number of such polygons is bounded by $\left(2^{O(\sqrt{\log n})}\right)^{O(L_2)} = 2^{O(\log^{3/2} n/\epsilon)} \le n^{O(\sqrt{\log n}/\epsilon)}$. The final set $\mathcal{C}$ of polygons corresponding to the entries of the dynamic programming table includes all level-1 polygons, and for every level-1 polygon P, all level-2 polygons in the set $\mathcal{C}(P)$. The algorithm for computing the entries of the dynamic programming table remains unchanged. This reduces the running time to $n^{O(\sqrt{\log n}/\epsilon^3)}$.

In order to improve the running time to $n^{\text{poly}(\log\log n/\epsilon)}$, we extend this approach to $O(\log\log n)$ recursive levels. As before, we partition the execution of the algorithm into phases, where a phase ends when the values of the optimal solutions of all instances involved in it decrease by a factor of roughly $\sqrt{n}$. Therefore, the algorithm has at most 2 phases. At the end of each phase, we employ a "clean-up" procedure, that reduces the number of corners of every polygon to $L_1 = \Theta((\log\log n)^3/\epsilon)$, with at most $2|\mathsf{OPT}| \cdot f(L_1)$ total loss in the number of rectangles from the optimal solution, where $f(L) = O(1/L)$. These polygons, that we call level-1 polygons, serve as the interface between the different phases. The set $\mathcal{C}_1$ of level-1 polygons then contains all polygons whose corners have integral coordinates between 1 and $2n$, that have at most L_1 corners. The number of such polygons is bounded by $n^{O(L_1)}$. Consider now an execution of a phase, and let P be our initial level-1 polygon. Since the values of the optimal solutions of instances considered in this phase are at least $|\mathsf{OPT}(\mathcal{R}(P))|/\sqrt{n}$, we can construct an $(O(\sqrt{n}) \times O(\sqrt{n}))$-grid G_P, that will serve as our discretization grid. We further partition the execution of the current phase (that we refer to as *level-1 phase*) into two level-2 phases, where the value of the optimal solution inside each level-2 phase goes down by a factor of roughly $n^{1/4}$, and we let $L_2 = 2L_1$. The total number of level-2 phases, across the execution of the whole algorithm, is then at most 4, and at the end of each such phase, we again apply a clean-up procedure, that decreases the number of corners in each polygon to L_2. The loss incurred in every level-2 phase due to the cleanup procedure can be bounded by $|\mathsf{OPT}| \cdot f(L_2) = |\mathsf{OPT}| \cdot f(L_1)/2$, and the total loss across all level-2 phases is at most $2|\mathsf{OPT}|f(L_1)$. For every level-1 polygon P, we define a set $\mathcal{C}_2(P)$ of level-2 polygons, that contains all polygons $P' \subseteq P$ with at most L_2 corners, so that P' is aligned with G_P. We continue the same procedure for $\Theta(\log\log n)$ recursive levels. For each level i, we let $L_i = 2L_{i-1} = 2^{i-1}L_1$, and we let $\rho_i = n^{1/2^i}$. In each level-i phase, the values of the optimal solutions to the instances defined by the corresponding polygons should decrease by a factor of roughly ρ_i, so there are approximately 2^i level-i phases overall. At the end of each phase, we apply the clean-up procedure, in order to decrease the number of corners of each polygon to L_i. The loss at the end of each level-i phase in the number of rectangles from the optimal solution is then at most $f(L_i) \cdot |\mathsf{OPT}| = f(L_1) \cdot |\mathsf{OPT}|/2^{i-1}$, and since the number of level-i phases is 2^i, the total loss due to the cleanup procedure in level-i phases is bounded by $2f(L_1)|\mathsf{OPT}|$. Summing up over all levels, the total loss due to the cleanup procedure is bounded by $2|\mathsf{OPT}|f(L_1)\log\log n \le \epsilon|\mathsf{OPT}|/\log\log n$. Additional loss is incurred due to the balanced $L_{\log\log n}$-corner partitions of level-$(\log\log n)$ instances, but this loss is analyzed as before, since $L_{\log\log n} = \Omega(\log n/\epsilon)$. In order to define level-i polygons, for every level-$(i-1)$ polygon P, we compute a grid G_P that has roughly $O(\rho_i)$ vertical and horizontal lines, so that for every vertical or horizontal strip S of the grid G_P, the value of the optimal solution of instance $\mathcal{R}(S \cap P)$ is roughy $|\mathsf{OPT}(\mathcal{R}(P))|/\rho_i$. We then let $\mathcal{C}_i(P)$ contain all polygons $P' \subseteq P$ that have at most L_i corners and are aligned with G_P. The final set of level-i polygons is the union of all sets $\mathcal{C}_i(P)$ for all level-$(i-1)$ polygons P. Let $\mathcal{C}_i$ denote the set of all level-i polygons, and let $\mathcal{C}$ be the set of all polygons of all levels. Then it is immediate to verify that $|\mathcal{C}_i| \le |\mathcal{C}_{i-1}| \cdot \rho_i^{O(L_i)} \le n^{O(L_1)}$, and since we employ $O(\log\log n)$ levels, overall $|\mathcal{C}| \le n^{O(L_1 \log\log n)} = n^{O((\log\log n)^4/\epsilon)}$.

Future directions. Unlike the algorithm of [1], our algorithm does not extend to the weighted setting where each rectangle has a weight and the goal is to find a maximum weight set of independent rectangles. The main technical obstacle is the discretization procedure where we construct a grid and we restrict the partitions into sub-instances to be aligned with the grid. In the weighted setting, there may be some heavy rectangles of the optimal solution that are not aligned with the grid. The optimal solution uses only a small number of such rectangles, but there may be many of them present in the input instance, and we do not know beforehand which of these rectangles are in the optimal solution and thus which rectangles to remove to obtain the alignment property. We leave the extension to the weighted setting, as well as more general shapes such as rectilinear polygons, as directions for future work.

Organization. We start with preliminaries in Section II, and summarize the discretization and partitioning theorems that we use in Section III. We then give a general outline of the dynamic programming–based algorithm and its analysis that we employ throughout the paper in Section IV. Section V contains a recap of the algorithm of Adamaszek and Wiese [1] in our framework; we also improve its running time to $n^{O(\log n/\epsilon^3)}$. In Section VI we show a QPTAS with running time $n^{O(\sqrt{\log n})/\epsilon^3}$, using the two-level approach. This approach is then extended in Section VII to $O(\log\log n)$ recursive levels, completing the proof of Theorem I.1. Due to lack of space, most of the proofs are omitted and can be found in the full version of the paper [9].

II. Preliminaries

In the Maximum Independent Set of Rectangles (MISR) problem, the input is a set $\mathcal{R} = \{R_1, R_2, ..., R_n\}$ of n open axis-parallel rectangles in the 2-dimensional plane. We say that two rectangles R_i and R_j intersect if $R_i \cap R_j \neq \emptyset$, and we say that they are disjoint otherwise. The goal in the MISR problem is to find a maximum-cardinality subset $\mathcal{R}^* \subseteq \mathcal{R}$ of rectangles, such that all rectangles in $\mathcal{R}^*$ are mutually disjoint.

Canonical Instances. We say that a set $\mathcal{R}$ of rectangles is *non-degenerate*, iff for every pair $R, R' \in \mathcal{R}$ of distinct rectangles, for every corner $p = (x, y)$ of R and every corner $p' = (x', y')$ of R', $x \neq x'$ and $y \neq y'$. We say that an input $\mathcal{R}$ to the MISR problem is *canonical*, iff $\mathcal{R}$ is a non-degenerate set of rectangles, whose corners have integral coordinates between 1 and $2n$. Using standard techniques, we can transform any input instance of the MISR problem into an equivalent canonical instance (see full version of the paper for more details). Therefore, we assume from now on that our input instance $\mathcal{R}$ is a canonical one. Let B be the rectangle whose lower left corner is $(0, 0)$ and upper right corner is $(2n + 1, 2n + 1)$. We call B the *bounding box* of $\mathcal{R}$.

Sub-Instances. Over the course of our algorithm, we will define sub-instances of the input instance $\mathcal{R}$. Each such sub-instance is given by some (not necessarily connected) region $S \subseteq B$, and it consists of all rectangles $R \in \mathcal{R}$ with $R \subseteq S$. In [1], each such sub-instance was given by a polygon S, whose boundary edges are parallel to the axes, and the boundary contains at most $\operatorname{poly}\log n/\epsilon$ edges. We define our sub-instances slightly differently. Each sub-instance is defined by a family $\mathcal{F}$ of at most $O(\log n/\epsilon)$ axis-parallel closed rectangles that are contained in B (but they do not necessarily belong to $\mathcal{R}$), where every pair of rectangles in $\mathcal{F}$ are mutually internally disjoint (that is, they are disjoint except for possibly sharing points on their boundaries). We call such rectangles $F \in \mathcal{F}$ *fake rectangles*. Let $S(\mathcal{F}) = B \setminus \left(\bigcup_{F\in\mathcal{F}} F\right)$. The sub-instance associated with $\mathcal{F}$, that we denote by $\mathcal{R}(\mathcal{F})$, consists of all rectangles $R \in \mathcal{R}$ with $R \subseteq S(\mathcal{F})$. In other words, we view the fake rectangles as "holes" in the area defined by the bounding box B, and we only consider input rectangles that do not intersect these holes. Given a sub-instance $\mathcal{R}(\mathcal{F})$, we denote the optimal solution to this sub-instance by $\mathsf{OPT}_{\mathcal{F}}$. The *boundary complexity* of the sub-instance $\mathcal{R}(\mathcal{F})$ is the number of the fake rectangles, $|\mathcal{F}|$. We denote by $\mathsf{OPT} = \mathsf{OPT}_{\emptyset}$ the optimal solution to the original problem. For convenience, throughout the paper, we denote by N the smallest integral power of 2 greater than $|\mathsf{OPT}|$, so $N = \Theta(|\mathsf{OPT}|)$. We say that a set $\mathcal{F}$ of fake rectangles is a *valid set of fake rectangles* iff $\mathcal{F}$ consists of closed rectangles that are mutually internally disjoint, and contained in B.

An $O(\log\log N)$-Approximation Algorithm. We need an algorithm that can estimate the values of the optimal solutions to various sub-instances of our problem instance. In [7], an $O(\log\log n)$-approximation algorithm was shown for MISR. This algorithm can be extended to achieve an $O(\log\log N)$-approximation; we defer the details to the full version of the paper. We denote this algorithm by $\mathcal{A}$, and, given any valid set $\mathcal{F}$ of fake rectangles, we denote by $\mathcal{A}(\mathcal{F})$ the value of the solution returned by this algorithm on instance $\mathcal{R}(\mathcal{F})$.

Decomposition Pairs and Triples. Our algorithm employs the Divide-and-Conquer paradigm, similarly to the algorithm of [1]. Intuitively, given a sub-instance $\mathcal{R}(\mathcal{F})$ of the problem, associated with the polygon $S(\mathcal{F})$, we would like to partition it into two (or sometimes three) sub-instances, associated with polygons S_1 and S_2, respectively. We require that $S_1 \cap S_2 = \emptyset$ and $S_1, S_2 \subsetneq S(\mathcal{F})$. Since we define the polygons in terms of the fake rectangles, we will employ the following definition.

Definition. *Let $\mathcal{F}$ any valid set of fake rectangles. We say that $(\mathcal{F}_1, \mathcal{F}_2)$ is a valid decomposition pair for $\mathcal{F}$ if for each $i \in \{1, 2\}$, $\mathcal{F}_i$ is a valid set of fake rectangles with $S(\mathcal{F}_i) \subsetneq S(\mathcal{F})$, and $S(\mathcal{F}_1) \cap S(\mathcal{F}_2) = \emptyset$. Similarly, we say that $(\mathcal{F}_1, \mathcal{F}_2, \mathcal{F}_3)$ is a valid decomposition triple for $\mathcal{F}$ if for each $1 \leq i \leq 3$, $\mathcal{F}_i$ is a valid set of fake rectangles with $S(\mathcal{F}_i) \subsetneq S(\mathcal{F})$, and for all $1 \leq i \neq j \leq 3$, $S(\mathcal{F}_i) \cap S(\mathcal{F}_j) = \emptyset$.*

III. Discretization and Partitioning

In this section we introduce the main technical tools used in our algorithms and their analysis. We start with grids that we use in order to discretize our instances.

A grid G of size $(z \times z)$ is defined by a collection $\mathcal{V} = \{V_0, \ldots, V_z\}$ of vertical lines, and a collection $\mathcal{H} = \{H_0, \ldots, H_z\}$ of horizontal lines. Each vertical line starts at the bottom of the bounding box and ends at the top of the bounding box, and we assume that $V_0, \ldots, V_z$ appear in this left-to-right order, where V_0 and V_z coincide with the left and the right boundaries of the bounding box B respectively. Similarly, each horizontal line starts at the left boundary and terminates at the right boundary of B, with $H_0, \dots, H_z$ appearing in this bottom-to-top order, where H_0, H_z coincide with the bottom and the top boundaries of B respectively. Each consecutive pair V_i, V_{i+1} of vertical lines defines a vertical strip S_i^V of the bounding box, and each consecutive pair H_j, H_{j+1} of horizontal lines defines a horizontal strip S_j^H.

Suppose we are given a grid G and a valid set $\mathcal{F}$ of fake rectangles. We say that $\mathcal{F}$ is *aligned* with G, iff every corner of every rectangle of $\mathcal{F}$ belongs to the set Z of the vertices of the grid G. We need the following two definitions of ρ-accurate grids, and a claim that allows us to construct them.

Definition. *Given a valid set $\mathcal{F}$ of fake rectangles and a parameter $\rho \geq 1$, we say that a grid $G = (\mathcal{V}, \mathcal{H})$ is ρ-accurate for $\mathcal{F}$, iff (i) for each vertical strip S_i^V, the value of the optimal solution of the sub-instance defined by all rectangles contained in $S_i^V \cap S(\mathcal{F})$ is at most $\lceil |\mathsf{OPT}_{\mathcal{F}}|/\rho \rceil$, and the same holds for each horizontal strip; and (ii) $\mathcal{F}$ is aligned with the grid G.*

Definition. *Let $G = (\mathcal{V}, \mathcal{H})$, $G' = (\mathcal{V}', \mathcal{H}')$ be two grids. We say that G' is* aligned *with G iff $\mathcal{V}' \subseteq \mathcal{V}$ and $\mathcal{H}' \subseteq \mathcal{H}$.*

Claim III.1 *Let $\mathcal{F}$ be any valid set of fake rectangles, and let G be a ρ-accurate grid for $\mathcal{F}$, for any parameter $\rho \geq 1$. Then for any $1 \leq \rho' \leq \rho$, we can efficiently construct a ρ'-accurate grid G' for $\mathcal{F}$ that is aligned with G, of size $(z \times z)$, where $z \leq O(\rho' \log\log(|\mathsf{OPT}_{\mathcal{F}}|) + |\mathcal{F}|)$.*

Finally, we state the main partitioning theorems used by our algorithms. These theorems extend and generalize similar theorems that were used in [1], [15].

Theorem III.2 *There is a universal constant $\tilde{c} > 10$, such that the following holds. For any parameter $L^* > \tilde{c}$, for any valid set $\mathcal{F}$ of fake rectangles, with $|\mathcal{F}| = L \leq L^*$ and $|\mathsf{OPT}_{\mathcal{F}}| \geq 512(L^*)^2$, given any ρ-accurate grid G for $\mathcal{F}$, where $\rho \geq 32(L^*)^2$, there is a valid decomposition triple $(\mathcal{F}_1, \mathcal{F}_2, \mathcal{F}_3)$ for $\mathcal{F}$, such that for all $1 \leq i \leq 3$, $|\mathcal{F}_i| \leq 3L^*/4$ and $|\mathsf{OPT}_{\mathcal{F}_i}| \leq 3|\mathsf{OPT}_{\mathcal{F}}|/4$. Moreover, $\sum_{i=1}^{3} |\mathsf{OPT}_{\mathcal{F}_i}| \geq |\mathsf{OPT}_{\mathcal{F}}| \cdot \left(1 - \frac{\tilde{c}}{L^*}\right)$, and the rectangles in $\mathcal{F}_1 \cup \mathcal{F}_2 \cup \mathcal{F}_3$ are aligned with the grid G.*

Theorem III.3 *For any valid set $\mathcal{F}$ of fake rectangles with $|\mathcal{F}| = L > \tilde{c}$ and $|\mathsf{OPT}_{\mathcal{F}}| \geq 512L^2$, given any ρ-accurate grid G for $\mathcal{F}$, where $\rho \geq 32L^2$, there is a valid decomposition pair $(\mathcal{F}_1, \mathcal{F}_2)$ for $\mathcal{F}$, such that $|\mathcal{F}_1|, |\mathcal{F}_2| \leq 3L/4$; $|\mathsf{OPT}_{\mathcal{F}_1}| + |\mathsf{OPT}_{\mathcal{F}_2}| \geq |\mathsf{OPT}_{\mathcal{F}}| \cdot \left(1 - \frac{\tilde{c}}{L}\right)$; and the rectangles in $\mathcal{F}_1 \cup \mathcal{F}_2$ are aligned with the grid G, where $\tilde{c}$ is the constant from Theorem III.2.*

IV. Algorithm Outline

All our algorithms follow the same general outline, that we describe here. We define a family $\mathcal{C}$ of *important sets of fake rectangles*, that contains $\emptyset$, so that every element of $\mathcal{C}$ is a valid set of fake rectangles. As an example, $\mathcal{C}$ may contain all valid sets $\mathcal{F}$ of fake rectangles with $|\mathcal{F}| \leq L^*$ for some bound L^*, such that all corners of all rectangles in $\mathcal{F}$ have integral coordinates. Additionally, we define a family $\mathcal{C}' \subseteq \mathcal{C}$ of *basic sets of fake rectangles*. Intuitively, for each $\mathcal{F} \in \mathcal{C}'$, the corresponding instance $\mathcal{R}(\mathcal{F})$ is "simple" in some sense. We assume that we are given an algorithm $\mathcal{A}'$, that, given a set $\mathcal{F} \in \mathcal{C}$ of fake rectangles, tests whether $\mathcal{F} \in \mathcal{C}'$, and an algorithm $\mathcal{A}''$ that can compute a $(1 - \epsilon/2)$-approximate solution to each instance $\mathcal{R}(\mathcal{F})$ with $\mathcal{F} \in \mathcal{C}'$. We discuss the running times of these algorithms later. We will ensure that $\{B\} \in \mathcal{C}'$ (the set of fake rectangles containing the bounding box only). This guarantees that for every set $\mathcal{F} \in \mathcal{C} \setminus \mathcal{C}'$ there is always a valid decomposition pair $(\mathcal{F}_1, \mathcal{F}_2)$ with $\mathcal{F}_1, \mathcal{F}_2 \in \mathcal{C}$ - for example, where $\mathcal{F}_1 = \mathcal{F}_2 = \{B\}$.

Once the families $\mathcal{C}, \mathcal{C}'$ of sets of fake rectangles, and algorithms $\mathcal{A}', \mathcal{A}''$ are fixed, our algorithm is also fixed, and proceeds via simple dynamic programming. The dynamic programming table T contains, for every important set $\mathcal{F} \in \mathcal{C}$ of fake rectangles, an entry $T[\mathcal{F}]$, that will store an approximate solution to the corresponding instance $\mathcal{R}(\mathcal{F})$. In order to initialize T, for every important set $\mathcal{F} \in \mathcal{C}$ of fake rectangles, we test whether $\mathcal{F} \in \mathcal{C}'$ using algorithm $\mathcal{A}'$, and if so, we apply algorithm $\mathcal{A}''$ to compute a valid $(1 - \epsilon/2)$-approximate solution to instance $\mathcal{R}(\mathcal{F})$, which is then stored at $T[\mathcal{F}]$. Once we finish the initialization step, we fill out the entries $T[\mathcal{F}]$ for $\mathcal{F} \in \mathcal{C} \setminus \mathcal{C}'$ from smaller to larger values of the area of $S(\mathcal{F})$.

Consider now some set $\mathcal{F} \in \mathcal{C} \setminus \mathcal{C}'$ of fake rectangles, and assume that for all $\mathcal{F}' \in \mathcal{C}$ with $S(\mathcal{F}') \subsetneq S(\mathcal{F})$, we have processed the entry $T[\mathcal{F}']$. Entry $T[\mathcal{F}]$ is computed as follows. For every triple $\mathcal{F}_1, \mathcal{F}_2, \mathcal{F}_3 \in \mathcal{C}$ of important sets of fake rectangles, such that $(\mathcal{F}_1, \mathcal{F}_2, \mathcal{F}_3)$ is a valid decomposition triple for $\mathcal{F}$, we consider the solution $\mathcal{X} = T[\mathcal{F}_1] \cup T[\mathcal{F}_2] \cup T[\mathcal{F}_3]$. We do the same for every pair $\mathcal{F}_1, \mathcal{F}_2 \in \mathcal{C}$ of important sets of fake rectangles, such that $(\mathcal{F}_1, \mathcal{F}_2)$ is a valid decomposition pair for $\mathcal{F}$. Among all such solutions $\mathcal{X}$, let $\mathcal{X}^*$ be the one of maximum value. We then store the solution $\mathcal{X}^*$ in $T[\mathcal{F}]$. Note that since we ensure that every set $\mathcal{F} \in \mathcal{C} \setminus \mathcal{C}'$ has a valid decomposition pair $(\mathcal{F}_1, \mathcal{F}_2)$ with $\mathcal{F}_1, \mathcal{F}_2 \in \mathcal{C}$, this step is well defined. This finishes the description of the algorithm. The final solution is stored in the entry $T[\emptyset]$. Notice that the choice of $\mathcal{C}, \mathcal{C}'$, and the algorithms $\mathcal{A}', \mathcal{A}''$ completely determine our algorithm. The running time depends on $|\mathcal{C}|$, the time required to construct $\mathcal{C}$, and the running times of the algorithms $\mathcal{A}'$ and $\mathcal{A}''$.

It is immediate to see that every entry $T[\mathcal{F}]$ of the dynamic programming table contains a feasible solution to instance $\mathcal{R}(\mathcal{F})$. We only need to show that the value of the solution stored in $T[\emptyset]$ is close to $|\mathsf{OPT}|$. This is done by constructing a *partitioning tree*, that we define below.

Definition. *Suppose we are given an instance $\mathcal{R}$ of the* MISR *problem, a family $\mathcal{C}$ of important sets of fake rectangles, and a subset $\mathcal{C}' \subseteq \mathcal{C}$ of basic sets of fake rectangles. Assume also that we are given a set $\mathcal{F} \in \mathcal{C}$ of fake rectangles. A partitioning tree $\mathcal{T}(\mathcal{F})$ for $\mathcal{F}$ is a tree, whose every vertex $v \in V(\mathcal{T})$ is labeled with a set $\mathcal{F}(v) \in \mathcal{C}$ of fake rectangles, such that: (i) if v is the root of the tree, then $\mathcal{F}(v) = \mathcal{F}$; and (ii) if v is an inner vertex of the tree, and $\{v_1, \ldots, v_r\}$ are its children, then $r \in \{2, 3\}$, and $\{\mathcal{F}(v_i)\}_{i=1}^{r}$ is either a valid decomposition pair or a valid decomposition triple for $\mathcal{F}(v)$. We say that $\mathcal{T}(\mathcal{F})$ is a* complete partitioning tree *for $\mathcal{F}$, if additionally for every leaf vertex v of $\mathcal{T}(\mathcal{F})$, $\mathcal{F}(v) \in \mathcal{C}'$. The loss of the tree $\mathcal{T}(\mathcal{F})$, denoted by $\Lambda(\mathcal{T}(\mathcal{F}))$, is $|\mathsf{OPT}_{\mathcal{F}}| - \sum_{v \in \mathcal{L}(\mathcal{T}(\mathcal{F}))} |\mathsf{OPT}_{\mathcal{F}(v)}|$, where $\mathcal{L}(\mathcal{T}(\mathcal{F}))$ denotes the set of leaves of $\mathcal{T}(\mathcal{F})$.*

A *full partitioning tree* for instance $\mathcal{R}$ is a complete partitioning tree $\mathcal{T}(\mathcal{F})$ for $\mathcal{F} = \emptyset$. For every vertex v of $\mathcal{T}$, we associate a value $\mu(v)$ with v, as follows. If v is a leaf of $\mathcal{T}$, then $\mu(v)$ is the value of the $(1 - \epsilon/2)$-approximate solution to instance $\mathcal{R}(\mathcal{F}(v))$ computed by the algorithm $\mathcal{A}''$. If v is an inner vertex of $\mathcal{T}$, then $\mu(v)$ is the sum of values $\mu(v')$ for all children v' of v. We denote by $\mu(\mathcal{T})$ the value $\mu(v)$ of the root vertex v of $\mathcal{T}$. The following observation connects the value of the solution computed by the dynamic programming algorithm to $\mu(\mathcal{T})$.

Observation IV.1 *For every vertex v of the full partitioning tree $\mathcal{T}$, the entry $T[\mathcal{F}(v)]$ of the dynamic programming table contains a solution to instance $\mathcal{R}(\mathcal{F}(v))$, whose value is at least $\mu(v)$.*

We use the following observation to analyze the approximation factors achieved by our algorithms.

Observation IV.2 *Suppose we are given an instance $\mathcal{R}$ of the* MISR *problem, a family $\mathcal{C}$ of important sets of fake rectangles, and a subset $\mathcal{C}' \subseteq \mathcal{C}$. Assume further that there exists a full partitioning tree $\mathcal{T}$ for $\mathcal{R}$, whose loss $\Lambda(\mathcal{T}) \leq \epsilon|\mathsf{OPT}|/2$. Then the dynamic programming-based algorithm described above computes a $(1-\epsilon)$-approximate solution to $\mathcal{R}$.*

Notice that in order to analyze our algorithm, we now only need to show the existence of a suitable partitioning tree. We do not need to provide an efficient algorithm to construct such a tree, and so we can assume that we know the optimal solution OPT to our instance $\mathcal{R}$ when constructing the tree.

V. A QPTAS WITH RUNNING TIME $n^{O(\log N/\epsilon^3)}$

In this section, we use the approach described in Section IV to obtain a QPTAS with running time $n^{O(\log N/\epsilon^3)}$. Let $L^* = 2\tilde{c}\log N/\epsilon$, where $\tilde{c}$ is the constant from Theorem III.2, and let $\tau = 512\,(L^*)^2 = \Theta(\log^2 N/\epsilon^2)$. Let $\rho^* = N$, and let G be the $((2n+1) \times (2n+1))$-grid, whose vertical and horizontal lines correspond to all integral x- and y-coordinates, respectively, between 0 and $2n+1$. Since we assumed that our instance $\mathcal{R}$ is canonical, it is immediate to verify that G is a ρ^*-accurate grid for set $\mathcal{F} = \emptyset$. Moreover, for any valid set $\mathcal{F}'$ of fake rectangles aligned with G, grid G remains ρ-accurate for $\mathcal{F}'$, for $\rho = |\mathsf{OPT}_{\mathcal{F}'}|$.

We let the family $\mathcal{C}$ of important sets of fake rectangles contain all valid sets $\mathcal{F}$ of fake rectangles with $|\mathcal{F}| \leq L^*$, such that all rectangles in $\mathcal{F}$ are aligned with G. Notice that any set $\mathcal{F} \in \mathcal{C}$ with $|\mathsf{OPT}_{\mathcal{F}}| > \tau$ and $|\mathcal{F}| \leq L^*$ satisfies the conditions of Theorem III.2 for grid G and value $\rho = |\mathsf{OPT}_{\mathcal{F}}|$.

It is immediate to verify that $|\mathcal{C}| = n^{O(L^*)} = n^{O(\log N/\epsilon)}$. The family $\mathcal{C}' \subseteq \mathcal{C}$ of basic sets of fake rectangles contains all sets $\mathcal{F}$ with $|\mathsf{OPT}_{\mathcal{F}}| \leq \tau$. We can verify whether $\mathcal{F} \in \mathcal{C}'$, and if so, we can find the optimal solution to instance $\mathcal{R}(\mathcal{F})$ in time $n^{O(\tau)} = n^{O(\log^2 N/\epsilon^2)}$ via exhaustive search. This defines the algorithms $\mathcal{A}'$ and $\mathcal{A}''$, and we can now employ the dynamic programming-based algorithm, described in Section IV. In order to analyze the running time of the algorithm, observe that the initialization step takes time $O(|\mathcal{C}|) \cdot n^{O(\log^2 N/\epsilon^2)} = n^{O(\log^2 N/\epsilon^2)}$, and the remaining part of the algorithm takes time $O(|\mathcal{C}|^4 \mathrm{poly}(n)) = n^{O(\log N/\epsilon)}$, so overall the running time is $n^{O(\log^2 N/\epsilon^2)}$. We later show how to improve the running time to $n^{O(\log N/\epsilon^3)}$ by replacing algorithms $\mathcal{A}'$ and $\mathcal{A}''$ with more efficient algorithms.

It now remains to show that the value of the solution computed by the algorithm is at least $(1-\epsilon)|\mathsf{OPT}|$. We do so by the constructing a full partitioning tree $\mathcal{T}$ for $\mathcal{R}$. We start with the tree $\mathcal{T}$ containing a single vertex v, with $\mathcal{F}(v) = \emptyset$. While there is a leaf vertex $v \in \mathcal{T}$ with $\mathcal{F}(v) \in \mathcal{C} \setminus \mathcal{C}'$ we add three children v_1, v_2 and v_3 to vertex v. Applying Theorem III.2 to $\mathcal{F}(v)$ with the grid G, we obtain a decomposition triple $(\mathcal{F}_1, \mathcal{F}_2, \mathcal{F}_3)$ for $\mathcal{F}(v)$, and we associate each of the three new vertices v_1, v_2, v_3 with the sets $\mathcal{F}_1, \mathcal{F}_2$ and $\mathcal{F}_3$, respectively. Notice that for $i \in \{1, 2, 3\}$, $|\mathcal{F}_i| \leq L^*$, and if $\mathcal{F}(v) \in \mathcal{C}$, then all rectangles in $\mathcal{F}_i$ are aligned with G, so $\mathcal{F}_1, \mathcal{F}_2, \mathcal{F}_3 \in \mathcal{C}$. The algorithm terminates when for every leaf vertex v of $\mathcal{T}$, $\mathcal{F}(v) \in \mathcal{C}'$. We then prove that the loss of the tree $\mathcal{T}$ is at most $\epsilon|\mathsf{OPT}|/2$. We defer the proofs to the full version of the paper.

So far, we have shown an algorithm that computes a $(1-\epsilon)$-approximate solution in time $n^{O(\log^2 N/\epsilon^2)}$. We can also use it to compute, for any instance $\mathcal{R}(\mathcal{F})$ defined by any valid set $\mathcal{F}$ of fake rectangles, a $(1-\epsilon/2)$-approximate solution for

$\mathcal{R}(\mathcal{F})$, in time $n^{O(\log^2|\mathsf{OPT}_{\mathcal{F}}|/\epsilon^2)}$. We denote this algorithm by $\mathcal{A}^*$.

We now describe a slightly modified version of the algorithm, whose running time becomes $n^{O(\log N/\epsilon^3)}$. We assume that $\epsilon > 1/\log^2 N$, since otherwise algorithm $\mathcal{A}^*$ provides an $(1-\epsilon)$-approximation in time $n^{O(\log N/\epsilon^3)}$. The family $\mathcal{C}$ of important sets of fake rectangles remains the same as before, but the family $\mathcal{C}' \subseteq \mathcal{C}$ of basic sets of fake rectangles is defined slightly differently: it contains all sets $\mathcal{F} \in \mathcal{C}$ of fake rectangles, such that the $O(\log\log N)$-approximation algorithm $\mathcal{A}$ returns a solution of value at most τ to instance $\mathcal{R}(\mathcal{F})$. We then let $\mathcal{A}'$ be the algorithm $\mathcal{A}$. Notice that if $\mathcal{F} \in \mathcal{C}'$, then $|\mathsf{OPT}_{\mathcal{F}}| \le O(\mathcal{A}(\mathcal{R}(\mathcal{F}))\log\log N) = O(\log^2 N \log\log N/\epsilon^2) = O(\operatorname{poly}\log N)$. We can now use algorithm $\mathcal{A}^*$ to compute a $(1-\epsilon/2)$-approximate solution for every instance $\mathcal{R}(\mathcal{F})$ with $\mathcal{F} \in \mathcal{C}'$, in time $n^{O(\log^2|\mathsf{OPT}_{\mathcal{F}}|/\epsilon^2)} = n^{O((\log\log N)^2/\epsilon^2)}$. The rest of the algorithm remains unchanged, except that we now use the algorithm $\mathcal{A}^*$ instead of $\mathcal{A}''$. It is immediate to verify that the running time of the algorithm becomes $n^{O(\log N/\epsilon^3)}$. For every set $\mathcal{F} \in \mathcal{C} \setminus \mathcal{C}'$ of fake rectangles, we now have $|\mathsf{OPT}_{\mathcal{F}}| \ge \mathcal{A}(\mathcal{F}) \ge \tau$, and so $\mathcal{F}$ is a valid input to Theorem III.2, together with grid G and $\rho = |\mathsf{OPT}_{\mathcal{F}}|$. We use the same construction of the partitioning tree as before, to show that the value of the solution returned by the algorithm is at least $(1-\epsilon)|\mathsf{OPT}|$.

VI. A QPTAS with Running Time $n^{O(\sqrt{\log N}/\epsilon^3)}$

In this section, we summarize our $(1-\epsilon)$-approximation algorithm with running time $n^{O(\sqrt{\log N}/\epsilon^3)}$. The algorithm follows the high-level overview provided in the Introduction, and it uses the framework introduced in Section IV. We use the following parameters: $L_1^* = \frac{100\tilde{c}\sqrt{\log N}}{\epsilon}$, $L_2^* = \frac{100\tilde{c}\log N}{\epsilon}$, and $\rho = \left(\frac{4}{3}\right)^{2\sqrt{\log N}}$, where $\tilde{c}$ is the parameter from Theorem III.2. All logarithms in this section are to the base of $4/3$. Let $\delta = \lceil \log(L_2^*/L_1^*)\rceil = \Theta(\log\log N)$, and let $\eta = 512(L_2^*)^{\delta+3}\cdot(4/3)^{2\sqrt{\log N}} = 2^{\Theta(\sqrt{\log N})}$. Let $\rho' = N$, and let G be the $((2n+1)\times(2n+1))$-grid, whose vertical and horizontal lines correspond to all integral x- and y-coordinates, respectively, between 0 and $2n+1$. It is easy to verify that G is ρ'-accurate for $\mathcal{F} = \emptyset$. The family $\mathcal{C}$ of important sets of fake rectangles is the union of two subsets, $\mathcal{C}_1$ and $\mathcal{C}_2$. Family $\mathcal{C}_1$ contains all valid sets $\mathcal{F}$ of fake rectangles, such that $|\mathcal{F}| \le L_1^*$, and all rectangles in $\mathcal{F}$ are aligned with G. Clearly, $|\mathcal{C}_1| = n^{O(\sqrt{\log N}/\epsilon)}$.

Consider now any important set $\mathcal{F} \in \mathcal{C}_1$ of fake rectangles. We define a collection $\mathcal{C}_2(\mathcal{F})$ of sets of fake rectangles, and we will eventually set $\mathcal{C}_2 = \bigcup_{\mathcal{F}\in\mathcal{C}_1}\mathcal{C}_2(\mathcal{F})$. In order to define the family $\mathcal{C}_2(\mathcal{F})$, we apply Claim III.1 to construct a ρ-accurate grid G' for $\mathcal{F}$, that is aligned with G. The size of the grid is $(z \times z)$, where $z = O(\rho\log\log N + |\mathcal{F}|) \le O(\rho\log\log N + \sqrt{\log N}/\epsilon) \le 2^{O(\sqrt{\log N})}$. We then let $\mathcal{C}_2(\mathcal{F})$ contain all valid sets $\mathcal{F}'$ of fake rectangles, with $S(\mathcal{F}') \subseteq S(\mathcal{F})$ and $|\mathcal{F}'| \le L_2^*$, such that $\mathcal{F}'$ is aligned with G. Clearly, $|\mathcal{C}_2(\mathcal{F})| \le z^{O(L_2^*)} = 2^{O(\log^{3/2}N/\epsilon)} = n^{O(\sqrt{\log N}/\epsilon)}$, and we can compute the family $\mathcal{C}_2(\mathcal{F})$ in time $n^{O(\sqrt{\log N}/\epsilon)}$. Finally, we set $\mathcal{C}_2 = \bigcup_{\mathcal{F}\in\mathcal{C}_1}\mathcal{C}_2(\mathcal{F})$, and $\mathcal{C} = \mathcal{C}_1 \cup \mathcal{C}_2$. Then $|\mathcal{C}_2| \le |\mathcal{C}_1| \cdot n^{O(\sqrt{\log N}/\epsilon)} \le n^{O(\sqrt{\log N}/\epsilon)}$ and thus $|\mathcal{C}| \le n^{O(\sqrt{\log N}/\epsilon)}$.

Recall that for a valid set $\mathcal{F}$ of fake rectangles, $\mathcal{A}(\mathcal{F})$ is the value of the solution returned by the $O(\log\log N)$-approximation algorithm $\mathcal{A}$ on input $\mathcal{R}(\mathcal{F})$. We now define the family $\mathcal{C}' \subseteq \mathcal{C}$ of basic sets of fake rectangles, and the corresponding algorithms $\mathcal{A}'$ and $\mathcal{A}''$. Family $\mathcal{C}'$ contains all sets $\mathcal{F} \in \mathcal{C}$ with $\mathcal{A}(\mathcal{F}) < \eta$, and we use the algorithm $\mathcal{A}$ in order to identify the sets $\mathcal{F} \in \mathcal{C}'$. Notice that if $\mathcal{F} \in \mathcal{C}'$, then $|\mathsf{OPT}_{\mathcal{F}}| \le O(\eta\log\log N) = 2^{O(\sqrt{\log N})}$. We can compute an $(1-\epsilon/2)$-approximate solution to each such instance $\mathcal{R}(\mathcal{F})$ in time $n^{O(\sqrt{\log N}/\epsilon^3)}$, using the algorithm from Section V, whose running time is $n^{O(\log|\mathsf{OPT}_{\mathcal{F}}|/\epsilon^3)} = n^{O(\sqrt{\log N}/\epsilon^3)}$. We employ this algorithm as $\mathcal{A}''$.

We can now use the dynamic programming-based algorithm from Section IV. The initialization step takes time at most $|\mathcal{C}| \cdot n^{O(\sqrt{\log N}/\epsilon^3)} = n^{O(\sqrt{\log N}/\epsilon^3)}$, and the rest of the algorithm runs in time $O(|\mathcal{C}|^4) = n^{O(\sqrt{\log N}/\epsilon)}$, so the total running time is $n^{O(\sqrt{\log N}/\epsilon^3)}$. It now remains to show that the algorithm computes a solution of value at least $(1-\epsilon)|\mathsf{OPT}|$. We do so using partitioning trees. We defer the details to the full version of the paper.

VII. A QPTAS with Running Time $n^{O((\log\log N)^4/\epsilon^4)}$

We start with an intuitive high-level overview of the algorithm. This overview is over-simplified and imprecise, and it is only intended to provide intuition. A natural way to further improve the running time of the QPTAS from Section VI is to use more levels of the recursion, namely: instead of just two sets $\mathcal{C}_1, \mathcal{C}_2 \subseteq \mathcal{C}$, we will have $h = \Theta(\log\log N)$ such sets, where we refer to the sets $\mathcal{F} \in \mathcal{C}_i$ as *level-i sets*, and to corresponding instances $\mathcal{R}(\mathcal{F})$ as *level-i instances*. We will also use parameters $L_1, \ldots, L_h$ associated with the instances of different levels. As before, family $\mathcal{C}_1$ will contain all valid sets $\mathcal{F}$ of fake rectangles, whose corners have integral coordinates, and $|\mathcal{F}| \le L_1$. For each $1 < i \le h$, for every set $\mathcal{F} \in \mathcal{C}_{i-1}$ of fake rectangles, we will define a family $\mathcal{C}_i(\mathcal{F})$ of sets of fake rectangles, as follows. We compute a ρ_{i-1}-accurate grid G_{i-1} for $\mathcal{F}$, for an appropriately chosen parameter ρ_{i-1}, and we let $\mathcal{C}_i(\mathcal{F})$ contain all valid sets $\mathcal{F}'$ of fake rectangles that are aligned with G_{i-1}, and have $|\mathcal{F}'| \le L_i$. We then set $\mathcal{C}_i = \bigcup_{\mathcal{F}\in\mathcal{C}_{i-1}}\mathcal{C}_i(\mathcal{F})$. Notice that the same set $\mathcal{F}'$ of fake rectangles may belong to several families $\mathcal{C}_i(\mathcal{F})$. It will be convenient in our analysis to view each such set as a separate set (though the algorithm does not distinguish between them), and to keep track of the sets of fake rectangles from $\mathcal{C}_1, \ldots, \mathcal{C}_{i-1}$, and their corresponding grids $G_1, \dots, G_{i-1}$, that were used to create the set $\mathcal{F}'$. In order to do so, we will denote each level-i instance by $\mathbb{F}^{(i)} = (\mathcal{F}_1, G_1, \dots, \mathcal{F}_{i-1}, G_{i-1}, \mathcal{F}_i)$, where for $1 \le i' < i$, $G_{i'}$ is a $\rho_{i'}$-accurate grid for $\mathcal{F}_{i'}$, though only the set $\mathcal{F}_i$ is added to $\mathcal{C}_i$.

All logarithms in this section are to the base of 2. For convenience of notation, we denote $\exp(i) = 2^i$. We assume that $\epsilon > 1/\log N$, since otherwise the $(1-\epsilon)$-approximation algorithm with running time $n^{O(\log N/\epsilon^3)}$ from Section V has running time $n^{O(1/\epsilon^4)}$.

We start with $h^* = \log\log N$. For each $1 \le i \le h^*$, we define a parameter $L_i = \frac{\tilde{c}\cdot(\log\log N)^3\cdot 2^i}{\epsilon}$, that will serve as the bound on the number of fake rectangles in each set $\mathcal{F} \in \mathcal{C}_i$. Notice that $L_1 < L_2 < \cdots < L_{h^*} = \frac{\tilde{c}\log N(\log\log N)^3}{\epsilon}$. We let $\eta = 32L_{h^*}^{2\delta+4}$. Since we have assumed that $\epsilon > 1/\log N$ and N is large enough, it is easy to verify that $\eta = \log^{\Theta(1)} N$.

For $0 \le i \le h^*$, we define $\rho_i = N^{1/2^i}$. Clearly, for all $1 \le i \le h^*$, $\rho_i = \sqrt{\rho_{i-1}}$. We let h be the largest integer, so that $\rho_h > \eta^{320}$. The number of the recursive levels in our construction will be h. Finally, we define the value $\tau^* = \rho_{h-1}^3 = (\log N)^{\Theta(1)}$.

Set $\mathcal{C}$ of important families of fake rectangles will eventually be a union of h subsets $\mathcal{C}_1, \dots, \mathcal{C}_h$. The execution of the algorithm at every level is partitioned into a number of phases. The optimal solution value in each phase of level i goes down by the factor of at least $(\rho_i)^{1/160}$. We then reduce the boundary complexities of the resulting level-$(i+1)$ instances from L_{i+1} to L_i.

Let G_0 be the $((2n+1)\times(2n+1))$-grid, whose vertical and horizontal lines correspond to all integral x- and y-coordinates, respectively, between 0 and $2n+1$. It is easy to see that G_0 is $\rho_0 = N$-accurate grid for $\mathcal{F} = \emptyset$. For all $1 \le i \le h$, it will be convenient to denote the level-i sets of fake rectangles by $\mathbb{F}^{(i)} = (\mathcal{F}_1, G_1, \dots, G_{i-1}, \mathcal{F}_i)$, where for $1 \le i' < i$, $G_{i'}$ is a $\rho_{i'}$-accurate grid for $\mathcal{F}_{i'}$, and all rectangles in $\mathcal{F}_{i'+1}$ are aligned with $G_{i'}$. We also require that for all $1 \le i' < i$, grid $G_{i'}$ is aligned with $G_{i'-1}$, and that for all $1 < i' \le i$, $S(\mathcal{F}_{i'}) \subseteq S(\mathcal{F}_{i'-1})$.

Level-1 Instances. We let $\mathcal{C}_1$ denote all valid sets $\mathcal{F}$ of fake rectangles with $|\mathcal{F}| \le L_1$, such that all rectangles in $\mathcal{F}$ are aligned with G_0. For each $\mathcal{F} \in \mathcal{C}_1$, we define the level-1 set $\mathbb{F}^{(1)} = (\mathcal{F})$ of fake rectangles to be consistent with our notation for higher-level sets. We denote $\tilde{\mathcal{C}}_1 = \left\{\mathbb{F}^{(1)} = (\mathcal{F}) \mid \mathcal{F} \in \mathcal{C}_1\right\}$. Notice that $|\mathcal{C}_1| \le n^{O(L_1)} = n^{O((\log\log N)^3/\epsilon)}$.

Level-i instances. Fix some $1 < i \le h$. For every level-$(i-1)$ instance $\mathbb{F}^{(i-1)} \in \tilde{\mathcal{C}}_{i-1}$, we define a set $\tilde{\mathcal{C}}_i(\mathbb{F}^{(i-1)})$ of level-i instances, and we let $\tilde{\mathcal{C}}_i = \bigcup_{\mathbb{F}^{(i-1)}\in\tilde{\mathcal{C}}_{i-1}} \tilde{\mathcal{C}}_i(\mathbb{F}^{(i-1)})$. We now describe the construction of the set $\tilde{\mathcal{C}}_i(\mathbb{F}^{(i-1)})$.

We assume that $\mathbb{F}^{(i-1)} = (\mathcal{F}_1, G_1, \mathcal{F}_2, G_2, \dots, G_{i-2}, \mathcal{F}_{i-1}) \in \tilde{\mathcal{C}}_{i-1}$ is a level-$(i-1)$ set of fake rectangles, where for each $1 \le i' < i-1$, $G_{i'}$ is a $\rho_{i'}$-accurate grid for $\mathcal{F}_{i'}$, and that grid $G_{i'}$ is aligned with grid $G_{i'-1}$. Moreover, for all $1 < i' \le i-1$, set $\mathcal{F}_{i'}$ contains at most $L_{i'}$ fake rectangles, that are aligned with the grid $G_{i'-1}$.

If $i > 2$ and $\mathcal{A}(\mathcal{F}_{i-1}) < \mathcal{A}(\mathcal{F}_{i-2})/\rho_{i-2}^{1/10}$, then we set $\tilde{\mathcal{C}}(\mathbb{F}^{(i)}) = \emptyset$. Assume now that $i > 2$ and $\mathcal{A}(\mathcal{F}_{i-1}) \ge \mathcal{A}(\mathcal{F}_{i-2})/\rho_{i-2}^{1/10}$. We use Claim III.1 to compute a ρ_{i-1}-accurate grid G_{i-1} for $\mathcal{F}_{i-1}$, so that G_{i-1} is aligned with G_{i-2}. If $i = 2$, then we simply compute any ρ_1-accurate grid G_1 for $\mathcal{F}_1$, that is aligned with G_0. In either case, the size of the grid is $(z \times z)$, where $z = O(\rho_{i-1}^2)$.

We construct the set $\tilde{\mathcal{C}}_i(\mathbb{F}^{(i-1)})$ as follows. For every valid set $\mathcal{F}'$ of fake rectangles, with $S(\mathcal{F}') \subseteq S(\mathcal{F}_{i-1})$, and $|\mathcal{F}'| \le L_i$, such that the rectangles in $\mathcal{F}'$ are aligned with the grid G_{i-1}, we add a level-i set $\mathbb{F}^{(i)} = (\mathcal{F}_1, G_1, \mathcal{F}_2, G_2, \dots, G_{i-2}, \mathcal{F}_{i-1}, G_{i-1}, \mathcal{F}')$ to $\tilde{\mathcal{C}}_i(\mathbb{F}^{(i-1)})$. Notice that $(\mathcal{F}_1, G_1, \dots, \mathcal{F}_{i-1}, G_{i-1}, \mathcal{F}_{i-1}) \in \tilde{\mathcal{C}}_i(\mathbb{F}^{(i-1)})$, and:

$$\begin{aligned}|\tilde{\mathcal{C}}_i(\mathbb{F}^{(i-1)})| &= z^{O(L_i)} \le \rho_i^{O(\exp(i)(\log\log N)^3/\epsilon)}\\ &= \left(N^{1/\exp(i)}\right)^{O(\exp(i)(\log\log N)^3/\epsilon)}\\ &= N^{O((\log\log N)^3/\epsilon)}.\end{aligned}$$

We set $\tilde{\mathcal{C}}_i = \bigcup_{\mathbb{F}^{(i-1)}\in\tilde{\mathcal{C}}_{i-1}} \tilde{\mathcal{C}}_i(\mathbb{F}^{(i-1)})$, and we let $\mathcal{C}_i$ contain all sets $\mathcal{F}$ of fake rectangles, such that for some $\mathbb{F}^{(i)} = (\mathcal{F}_1, G_1, \dots, G_{i-1}, \mathcal{F}_i) \in \tilde{\mathcal{C}}_i$, $\mathcal{F} = \mathcal{F}_i$. Finally, we set $\mathcal{C} = \bigcup_{i=1}^h \mathcal{C}_i$.

We say that $\mathcal{F} \in \mathcal{C}$ is a basic set of fake rectangles, and add it to $\mathcal{C}'$, iff $\mathcal{A}(\mathcal{F}) \le \tau^*$. We can use the algorithm $\mathcal{A}$ to determine, for each set $\mathcal{F} \in \mathcal{C}$, whether $\mathcal{F}$ is a basic set. If $\mathcal{F}$ is a basic set, then $|\mathsf{OPT}_{\mathcal{F}}| \le \mathcal{A}(\mathcal{F}) \cdot O(\log\log N) \le \tau^* \cdot O(\log\log N) = (\log N)^{O(1)}$, and we can use the algorithm from Section VI to compute a $(1-\epsilon/2)$-approximate solution to instance $\mathcal{R}(\mathcal{F})$ in time $n^{O(\sqrt{\log|\mathsf{OPT}_{\mathcal{F}}|}/\epsilon^3)} = n^{O(\log\log N/\epsilon^3)}$. We use this algorithm as algorithm $\mathcal{A}''$ for the initialization step of the dynamic program. This completes the definition of the family $\mathcal{C}$ of important sets of fake rectangles, the family $\mathcal{C}' \subseteq \mathcal{C}$ of basic sets of fake rectangles, and the algorithms $\mathcal{A}'$ and $\mathcal{A}''$. We then use the dynamic programming-based algorithm from Section IV to solve the problem. In order to analyze the running time of the algorithm, we first need to bound $|\mathcal{C}|$. As we showed above,

$$|\tilde{\mathcal{C}}_1| \le O\left(n^{L_1}\right) = n^{O((\log\log N)^3/\epsilon)},$$

and for all $1 < i \le h$,

$$|\tilde{\mathcal{C}}_i| = \sum_{\mathbb{F}^{(i-1)} \in \tilde{\mathcal{C}}_{i-1}} |\tilde{\mathcal{C}}(\mathbb{F}^{(i-1)})| \leq |\tilde{\mathcal{C}}_{i-1}| \cdot N^{O((\log\log N)^3/\epsilon)}.$$

Since $h < \log\log N$, it is immediate to verify that $|\mathcal{C}| = O(|\tilde{\mathcal{C}}_h|) \leq n^{O((\log\log N)^4/\epsilon)}$. The initialization step then takes time $|\mathcal{C}| \cdot n^{O(\log\log N/\epsilon^3)} = n^{O((\log\log N)^4/\epsilon^3)}$, and the remainder of the algorithm runs in time $|\mathcal{C}|^{O(1)}$. Therefore, the total running time of the algorithm is bounded by $n^{O((\log\log N)^4/\epsilon^3)}$.

It now remains to show that the algorithm computes a $(1-\epsilon)$-approximate solution. In order to do so, we construct a complete partitioning tree and show that its loss is suitably bounded.

The remaining details of the algorithm and its analysis appear in the full version of the paper [9].

Acknowledgements. The work of the first author is supported in part by NSF grant CCF-1318242. This work was done in part while the second author was with the Computer Science department at Princeton University and University of Warwick, and a visitor at the Simons Institute for the Theory of Computing and the Toyota Technological Institute at Chicago.

References

[1] Anna Adamaszek and Andreas Wiese. Approximation schemes for maximum weight independent set of rectangles. In *IEEE Foundations of Computer Science (FOCS)*, pages 400–409. IEEE Computer Society, 2013.

[2] Anna Adamaszek and Andreas Wiese. A QPTAS for maximum weight independent set of polygons with polylogarithmically many vertices. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 645–656. SIAM, 2014.

[3] Pankaj K Agarwal, Marc Van Kreveld, and Subhash Suri. Label placement by maximum independent set in rectangles. *Computational Geometry*, 11(3):209–218, 1998.

[4] Piotr Berman, Bhaskar DasGupta, S. Muthukrishnan, and Suneeta Ramaswami. Improved approximation algorithms for rectangle tiling and packing. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 427–436. SIAM, 2001.

[5] Ravi Boppana and Magnús M Halldórsson. Approximating maximum independent sets by excluding subgraphs. *BIT Numerical Mathematics*, 32(2):180–196, 1992.

[6] Parinya Chalermsook. Coloring and maximum independent set of rectangles. In *International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX)*, pages 123–134. Springer, 2011.

[7] Parinya Chalermsook and Julia Chuzhoy. Maximum independent set of rectangles. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 892–901. SIAM, 2009.

[8] Timothy M. Chan and Sariel Har-Peled. Approximation algorithms for maximum independent set of pseudo-disks. *Discrete & Computational Geometry*, 48(2):373–392, 2012.

[9] Julia Chuzhoy and Alina Ene. On approximating maximum independent set of rectangles. arXiv preprint 1608.00271, 2016.

[10] Jeffrey S Doerschler and Herbert Freeman. A rule-based system for dense-map name placement. *Communications of the ACM*, 35(1):68–79, 1992.

[11] Thomas Erlebach, Klaus Jansen, and Eike Seidel. Polynomial-time approximation schemes for geometric intersection graphs. *SIAM Journal on Computing (SICOMP)*, 34(6):1302–1323, 2005.

[12] 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.

[13] Jacob Fox and János Pach. Computing the independence number of intersection graphs. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1161–1165. SIAM, 2011.

[14] Takeshi Fukuda, Yasuhiko Morimoto, Shinichi Morishita, and Takeshi Tokuyama. Data mining with optimized two-dimensional association rules. *ACM Transactions on Database Systems (TODS)*, 26(2):179–213, 2001.

[15] Sariel Har-Peled. Quasi-polynomial time approximation scheme for sparse subsets of polygons. In *ACM Symposium on Computational Geometry (SoCG)*, page 120. ACM, 2014.

[16] Johan Håstad. Some optimal inapproximability results. *Journal of the ACM (JACM)*, 48(4):798–859, 2001.

[17] Hiroshi Imai and Takao Asano. Finding the connected components and a maximum clique of an intersection graph of rectangles in the plane. *Journal of Algorithms*, 4(4):310–323, 1983.

[18] Sanjeev Khanna, S. Muthukrishnan, and Mike Paterson. On approximating rectangle tiling and packing. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, volume 95, page 384. SIAM, 1998.

[19] Brian Lent, Arun Swami, and Jennifer Widom. Clustering association rules. In *International Conference on Data Engineering*, pages 220–231. IEEE, 1997.

[20] Liane Lewin-Eytan, Joseph Seffi Naor, and Ariel Orda. *Routing and admission control in networks with advance reservations*. Springer, 2002.

[21] Nabil H Mustafa, Raghu Raman, and Sambaran Ray. Settling the APX-hardness status for geometric set cover. In *IEEE Foundations of Computer Science (FOCS)*, pages 541–550. IEEE, 2014.

[22] Frank Nielsen. Fast stabbing of boxes in high dimensions. *Theoretical Computer Science*, 246(1):53–72, 2000.