On Rectilinear Partitions with Minimum Stabbing Number*

Mark de Berg[1], Amirali Khosravi[1], Sander Verdonschot[2], and Vincent van der Weele[3]

[1] TU Eindhoven, P.O. Box 513, 5600 MB Eindhoven, the Netherlands.
[2] School of Computer Science, Carleton University, Ottawa, Canada.
[3] Max-Planck-Institut für Informatik, Saarbrücken, Germany.

Abstract. Let S be a set of n points in $\mathbb{R}^d$, and let r be a parameter with $1 \leqslant r \leqslant n$. A rectilinear r-partition for S is a collection $\Psi(S) := \{(S_1, b_1), \ldots, (S_t, b_t)\}$, such that the sets S_i form a partition of S, each b_i is the bounding box of S_i, and $n/2r \leqslant |S_i| \leqslant 2n/r$ for all $1 \leqslant i \leqslant t$. The (rectilinear) stabbing number of $\Psi(S)$ is the maximum number of bounding boxes in $\Psi(S)$ that are intersected by an axis-parallel hyperplane h. We study the problem of finding an optimal rectilinear r-partition—a rectilinear partition with minimum stabbing number—for a given set S. We obtain the following results.

- Computing an optimal partition is NP-hard already in $\mathbb{R}^2$.
- There are point sets such that any partition with disjoint bounding boxes has stabbing number $\Omega(r^{1-1/d})$, while the optimal partition has stabbing number 2.
- An exact algorithm to compute optimal partitions, running in polynomial time if r is a constant, and a faster 2-approximation algorithm.
- An experimental investigation of various heuristics for computing rectilinear r-partitions in $\mathbb{R}^2$.

1 Introduction

Motivation. Range searching is one of the most fundamental problems in computational geometry. In its basic form it can be stated as follows: preprocess a set S of objects in $\mathbb{R}^d$ into a data structure such that the objects intersecting a query range can be reported (or counted) efficiently. The range-searching problem has many variants, depending for example on the type of objects (points, or some other type of objects), on the dimension of the underlying space (two- or higher dimensional), and on the type of query range (boxes, simplices, etc.)—see the survey of Agarwal and Erickson [1] for an overview.

* This research was supported by the Netherlands' Organisation for Scientific Research (NWO) under project no. 639.023.301 and project no. 612.000.631. *Note to reviewers:* Part of this research was presented at the 26th European Workshop on Computational Geometry (2010). EuroCG is an informal workshop without formal proceedings, and it is customary that EuroCG papers are also submitted to peer-reviewed conferences and/or journals. See www.eurocg.org/about.html.

A range-searching data structure that is popular in practice is the *bounding-volume hierarchy*, or BVH for short. This is a tree in which each object from S is stored in a leaf, and each internal node stores a bounding volume of the objects in its subtree. Often the bounding volume is a *bounding box*: the smallest axis-aligned box containing the objects in the subtree. When a BVH is stored in external memory, one usually uses a B-tree [5, Chapter 18] as underlying tree structure; the resulting structure (with bounding boxes as bounding volumes) is then called an *R-tree*. R-trees are one of the most widely used external-memory data structures for spatial data, and they have been studied extensively—see for example the book by Manolopoulos *et al.* [11]. In this paper we study a fundamental problem related to the construction of R-trees, as explained next.

One common strategy to construct R-trees is the top-down construction. Top-down construction algorithms partition S into a number of subsets S_i, and then recursively construct a subtree $\mathcal{T}_i$ for each S_i. Thus the number of subsets corresponds to the degree of the R-tree. When a range query with a range Q is performed, one has to recursively search in the subtrees $\mathcal{T}_i$ for which the bounding box of S_i (denoted b_i) intersects Q. If $b_i \subset Q$, then all objects stored in $\mathcal{T}_i$ lie inside Q; if, however, b_i intersects ∂Q (the boundary of Q) then we do not know if the objects stored in $\mathcal{T}_i$ intersect Q. Thus the overhead of the search algorithm is determined by the bounding boxes intersecting ∂Q. If Q is a box, as is often the case, then the number of bounding boxes b_i intersecting ∂Q is bounded, up to a factor $2d$, by the maximum number of bounding boxes intersecting any axis-parallel plane. Thus we want to partition S into subsets so as to minimize the number of bounding boxes intersecting any axis-parallel plane.

Further background and problem statement. Let S be a set of n points in $\mathbb{R}^d$, and let r be a parameter with $1 \leqslant r \leqslant n$. A *rectilinear r-partition* for S is a collection $\Psi(S) := \{(S_1, b_1), \ldots, (S_t, b_t)\}$ such that the sets S_i form a partition of S, each b_i is the bounding box of S_i, and $n/2r \leqslant |S_i| \leqslant 2n/r$ for all $1 \leqslant i \leqslant t$. Note that even though the subsets S_i form a (disjoint) partition of S, the bounding boxes b_i need not be disjoint. The *stabbing number* of an axis-parallel hyperplane h with respect to $\Psi(S)$ is the number of boxes b_i whose relative interior intersects h, and the *(rectilinear) stabbing number* of $\Psi(S)$ is the maximum stabbing number of any axis-parallel plane h. Observe that our rectilinear r-partitions are the axis-parallel counterpart of the (fine) simplicial partitions introduced by Matoušek [12].

It has been shown that there are point sets S for which any rectilinear r-partition has stabbing number $\Omega(r^{1-1/d})$ [12]; an example is the case when the points in S form a grid of size $n^{1/d} \times \cdots \times n^{1/d}$. Moreover, any set S admits a rectilinear r-partition with stabbing number $O(r^{1-1/d})$; such a rectilinear r-partition can be obtained by a construction similar to a kd-tree [6]. Thus from a worst-case and asymptotic point of view the problem of computing rectilinear r-partitions with low stabbing number is solved. However, any specific point set may admit a rectilinear r-partition with a much lower stabbing number than $\Theta(r^{1-1/d})$. For instance, if the points from S are all collinear *on a diagonal line*, then there is a rectilinear r-partition with stabbing number 1. Note that in this case the rectangles are axis-parallel segments, and their stabbing number in one of the directions is 0. The question now arises: given a point set S and a parameter r, can we compute a rectilinear r-partition that is *optimal for the given input set S*, rather than worst-case optimal? In other words, we want to compute a rectilinear r-partition that has the minimum stabbing number over all rectilinear r-partitions for S.

Our results. We start by a theoretical investigation of the complexity of the problem of finding optimal rectilinear r-partitions. In Section 2 we show that already in $\mathbb{R}^2$, finding an optimal rectilinear r-partition is NP-hard if r is considered as a parameter. In Section 3 we then give an exact algorithm for computing optimal rectilinear r-partitions which runs in polynomial time if r is a constant, and a 2-approximation with a better running time. We conclude our theoretical investigations by showing that algorithms only considering partitions with disjoint bounding boxes cannot have a good approximation ratio: there are point sets such that any partition with disjoint bounding boxes has stabbing number $\Omega(r^{1-1/d})$, while the optimal partition has stabbing number 2. We also perform an experimental investigation of various heuristics for computing rectilinear r-partitions with small stabbing number in $\mathbb{R}^2$. A simple variant of a kd-tree approach, which we call the *windmill kd-tree* turns out to give the best results.

2 Finding optimal rectilinear r-partitions is NP-hard

The exact problem we consider in this section is as follows.

Optimal Rectilinear r-Partition
Input: A set S of n points in $\mathbb{R}^2$ and two parameters r and k.
Output: YES if S admits a rectilinear r-partition w.r.t. r with stabbing number at most k, NO otherwise.

We will show that this problem is already NP-complete for fixed values of k.

Theorem 1. Optimal Rectilinear r-Partition *is* NP*-complete for* $k = 5$.

To prove the theorem we use a reduction from 3-SAT, which is similar to the proof by Fekete *et al.* [8] of the NP-hardness of minimizing the stabbing number of a matching on a planar point set. Let $\mathcal{U} := \{x_1, \ldots, x_m\}$ be a set of m boolean variables, and let $\mathcal{C} := C_1 \wedge \cdots \wedge C_s$ be a CNF formula defined over these variables, where each clause C_i is the disjunction of three variables. The 3-SAT problem is to decide whether such a boolean formula is satisfiable; 3-SAT is NP-hard [9]. Our reduction will be such that there is a rectilinear r-partition with stabbing number $k = 5$ for the Optimal Rectilinear r-Partition instance if and only if the 3-SAT instance is satisfiable. To simplify the reduction we assume that $n = 72r$ (to make $\sqrt{2n/r}$ an integer greater than or equal to 12); however, the reduction works for any $n = \alpha \cdot r$ for $\alpha \geqslant 72$. We first describe the various gadgets we need and then explain how to put them together.

The barrier gadget. A barrier gadget is a set G of $25 \cdot h^2$ points, where $h \geqslant 12$ and $h^2 = 2n/r$, arranged in a regular $5h \times 5h$ grid. To simplify the construction we fix

$h = 12$. Thus a barrier gadget is simply a 60×60 grid placed in a small square. The idea is essentially that if we partition a barrier gadget and require stabbing number 5, then both the vertical and the horizontal stabbing numbers will be 5. This will prevent any other bounding boxes from crossing the vertical strip (and, similarly, the horizontal strip) whose bounding lines contain the vertical (resp. horizontal) edges of the square containing the barrier gadget. Thus the barrier gadget can be used to make sure there is no interaction between different parts of the global construction. Lemma 1 below makes this precise by giving a bound on the minimum stabbing number of any r-partition of a barrier gadget. In fact, we are interested in the case where G is a subset of a larger set S. In our construction we will place any barrier gadget G in such a way that the points in $S \setminus G$ lie outside the bounding box of G, so when analyzing the stabbing number of a barrier gadget we will always assume that this is the case.

Let G be a barrier gadget and $S \supset G$ be a set of n points. We define $\Psi(S \downarrow G)$, the *restriction to* G of a rectilinear r-partition $\Psi(S) = \{(S_1, b_1), \ldots, (S_t, b_t)\}$, as

$$\Psi(S \downarrow G) := \{(S_i \cap G, b_i) : 1 \leqslant i \leqslant t \text{ and } S_i \cap G \neq \emptyset\}.$$

In other words, the boxes in $\Psi(S \downarrow G)$ are the boxes from $\Psi(S)$ whose associated point set contains at least one point from the barrier. The following lemma, which is proved in the appendix, gives a bound on the vertical and horizontal stabbing number of a rectilinear partition of a barrier gadget, where the vertical (horizontal) stabbing number is defined as the maximum number of boxes intersected by any vertical (horizontal) line.

Lemma 1. *A barrier gadget G can be covered by a set of 25 boxes with stabbing number 5. Moreover, for any rectilinear r-partition $\Psi(S)$ of stabbing number 5, the restriction $\Psi(S \downarrow G)$ has vertical as well as horizontal stabbing number 5.*

The variable gadget. Fig. 1 shows the variable gadget. The three subsets in the left part of the construction, and the three subsets in the right part, each contain $n/2r = 36$ points. Because of the barrier gadgets, the points from one subset cannot be combined with other points and must be put together into one rectangle in the partition. The six subsets in the middle part of the construction each contain $4n/r = 288$ points. To make sure the stabbing number does not exceed 5, these subsets can basically be grouped in two different ways. One grouping corresponds to setting the variable to true, the other grouping to false—see Fig. 1. Note that the gadget defines two vertical *slabs*. If the variable is set to true then the left slab has stabbing number 2 and the right slab has stabbing number 4, otherwise the opposite is the case.

The clause gadget. A clause gadget consists of three subsets of $4n/r = 288$ points, arranged as shown in Fig. 2(a), and placed in the left or right slab of the corresponding variables: a positive literal is placed in the left slab, a negative lateral in the right slab. If the stabbing number of the slab is already 4, which is the case when the literal evaluates to false, then the subset of $4n/r$ points in the clause gadget must be grouped into two "vertical" rectangles. Hence, not all literals in a clause can evaluate to false if the stabbing number is to be at most 5.

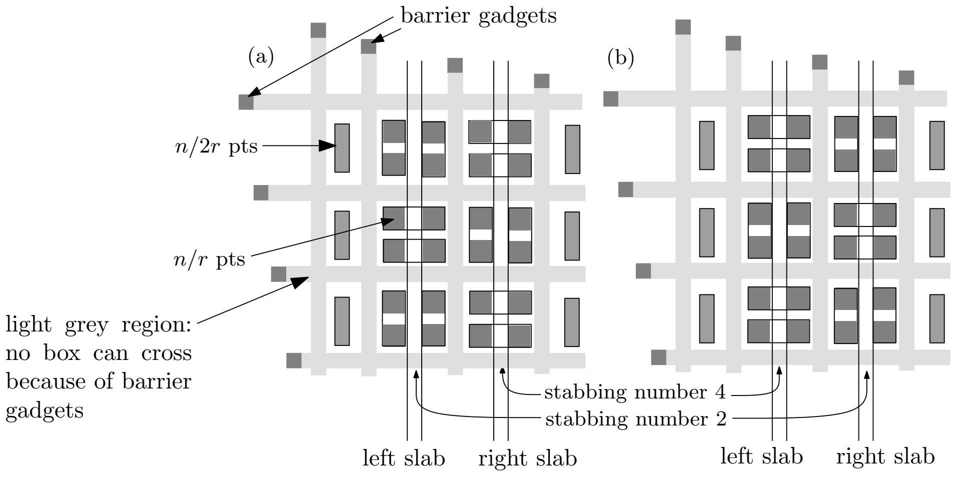

Fig. 1. The variable gadget. (a) True setting. (b) False setting.

The global structure. The global construction is shown in Fig. 2(b). There are variable gadgets, clause gadgets, and barrier gadgets. The variable gadgets are placed diagonally and the clause gadgets are placed below the variables. We also place barriers separating the clause gadgets from each other. Finally, the gadgets for occurrences of the same variable in different clauses should be placed such that they are not stabbed by a common vertical line. This concludes our sketch of the construction which proves Theorem 1. A formal proof of the theorem can be found in the appendix.

3 Polynomial time algorithms for constant r

In the previous section we showed that Optimal Rectilinear r-Partition is NP-hard when r is considered part of the input. Now we give a simple algorithm

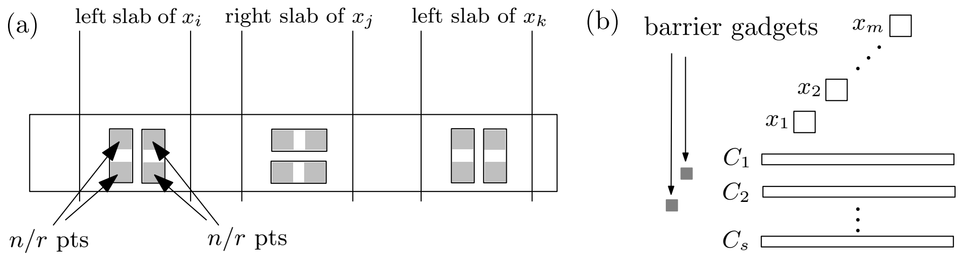

Fig. 2. (a) A clause gadget for $(x_i \vee \overline{x_j} \vee x_k)$, and one possible grouping of the points. (b) The global structure.

to show that the problem *in* $\mathbb{R}^d$ can be solved in polynomial time for fixed r in dimension d. Our algorithm works as follows.

1. Let C be the set of all boxes defined by at most $2d$ points in S. Note that $|C| = O(n^{2d})$.
2. For each t with $r/2 \leqslant t \leqslant 2r$, proceed as follows. Consider all $O(n^{2dt})$ possible subsets $B \subset C$ with $|B| = t$. Check whether B induces a valid solution, that is, whether we can assign the points in S to the boxes in B such that (i) each point is assigned to a box containing it, and (ii) each box is assigned between $n/2r$ and $2n/r$ points. How this is done will be explained later.
3. Over all sets B that induce a valid solution, take the set with the smallest stabbing number. Replace each box in it with the bounding box of the points assigned to it, and report the partition.

To implement Step 2 we construct a flow network with node set $\{v_{\text{source}}, v_{\text{sink}}\} \cup S \cup B$. The source node v_{source} has an arc of capacity 1 to each point $p \in S$, each $p \in S$ has an arc of capacity 1 to every $b_j \in B$ that contains p, and each $b_j \in B$ has an arc of capacity $2n/r$ to the sink node v_{sink}. The arcs from the boxes to the sink also have (besides the upper bound of $2n/r$ on the flow) a lower bound of $n/2r$ on the flow. The set B induces a valid rectilinear r-partition if and only if there is an integer flow of n units from v_{source} to v_{sink}. Such a flow problem can be solved in $O(\min(V^{3/2}, E^{1/2})E \log(V^2/E + 2) \log c)$ time [2], where V is the number of vertices in the network, E is the number of arcs, and c is the maximum capacity of any arc. We have $V = O(n)$, $E = O(nr)$, and $c = 2n/r$. Since we have to check $O(n^{4dr})$ subsets B, the running time is $O(n^{4dr} \cdot (nr)^{3/2} \log^2(n/r + 2))$ and is polynomial (assuming r is a constant). We obtain the following result.

Theorem 2. *Let S be a set of n points in $\mathbb{R}^d$, and r a constant. Then we can compute a rectilinear r-partition with optimal stabbing number in time $O(n^{4dr+3/2} \log^2 n)$*

We can significantly improve the running time if we are satisfied with a 2-approximation. The trick is to place a collection H_i of $3r$ hyperplanes orthogonal to the x_i-axis (the ith-axis) such that there are at most $n/3r$ points from S in between any two consecutive hyperplanes in H_i. Instead of finding $O(n^{2d})$ boxes in the first step of the algorithm, we now find $O(r^{2d})$ boxes defined by the hyperplanes in $H := H_1 \cup \cdots \cup H_d$. Then we have $|C| = O(r^{2d})$. We apply the Step 2 of the algorithm and find for $r/2 \leqslant t \leqslant 2r$ all the $O(r^{2dt})$ subsets $B \subset C$. We check for each subset whether it is a valid solution, and take the best valid solution. Lemma 2 in the appendix shows this gives a 2-approximation.

Theorem 3. *Let S be a set of n points, and r a constant. Then we can compute a rectilinear r-partition with stabbing number at most $2 \cdot \text{OPT}$, where OPT is the minimum stabbing number of any rectilinear partition for S, in time $O(n^{3/2} \log^2 n)$.*

4 Arbitrary versus disjoint rectilinear r-partitions

Since computing optimal rectilinear r-partitions is NP-hard, one should look at approximation algorithms. It may be easier to develop an approximation

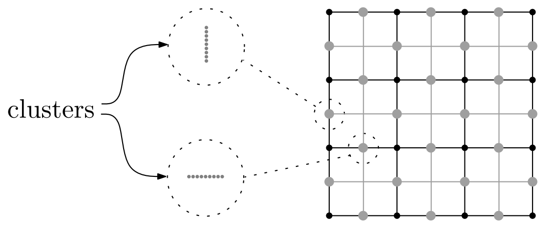

Fig. 3. Every rectilinear r-partition with disjoint bounding boxes has stabbing number $\Omega(\sqrt{r})$ while there exists a partition with stabbing number 2.

algorithm considering only rectilinear r-partitions with disjoint bounding boxes. The next theorem shows that in $\mathbb{R}^2$ such an approach will not give a good approximation ratio. The same argument holds for $\mathbb{R}^d$. (see the appendix)

Theorem 4. *Assume that $32 \leqslant r \leqslant 4 \cdot \sqrt{n}$. Then there is a set S of n points in $\mathbb{R}^2$ whose optimal rectilinear r-partition has stabbing number 2, while any rectilinear r-partition with disjoint bounding boxes has stabbing number $\Omega(\sqrt{r})$.*

Proof. Let $\mathcal{G}$ be a $\sqrt{r/8} \times \sqrt{r/8}$ grid in $\mathbb{R}^2$. (For simplicity assume that $\sqrt{r/8}$ is an integer. Since $32 \leqslant r$ we have $\sqrt{r/8} \geqslant 2$.) We put each grid point in S and call them *black points*. We call the lines forming the grid $\mathcal{G}$ *black lines*. Note that there are $r/8$ black points. Fig. 3 shows an example with $r = 128$. Next we refine the grid using $2(\sqrt{r/8} - 1)$ additional axis-parallel *grey lines*. At each of the new grid points that is not fully defined by gray lines—the grey dots in the figure—we put a tiny cluster of $2n/r$ points, which we also put in S. If the cluster lies on one or more black lines, then all points from the cluster lie in the intersection of those lines, as shown in Fig. 3.

So far we used $(2r/8 - 2 \cdot \sqrt{r/8}) \cdot 2n/r + r/8$ points. Since $r \leqslant 4 \cdot \sqrt{n}$, the number of points which we used so far is less than n. The remaining points can be placed far enough from the construction (not influencing the coming argument.) Next, we rotate the whole construction slightly so that no two points have the same coordinate in any dimension. This rotated set is our final point set S.

To obtain a rectilinear r-partition with stabbing number 2, we make each of the clusters into a separate subset S_i, and put the black points into one separate subset; the latter is allowed since $r/8 \leqslant 2n/r$. (If $r/8 < n/2r$ we can use some of the remaining points or the points of gray dots to fill up the subset.)

If the clusters are small enough, then the rotation we have applied to the point set guarantees that no axis-parallel line can intersect two clusters at the same time. *Any line intersects at most one of the clusters and the rectangle containing the black points, and the stabbing number of this rectilinear r-partition is 2*.

We claim that any disjoint rectilinear r-partition for S has stabbing number $\Omega(\sqrt{r/8})$. To see this, observe that no subset S_i in a disjoint rectilinear r-partition can contain two black points. Indeed, the bounding box of any two black points contains at least one full cluster and, hence, together with the black points would

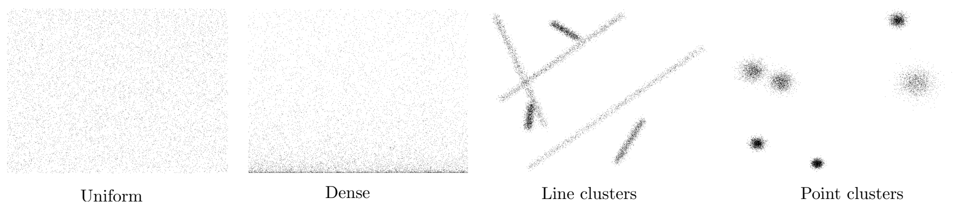

Fig. 4. The different types of input sets.

be too many points. We conclude that each black point is assigned to a different bounding box. Let B be the collection of these bounding boxes. Now duplicate each of the black lines, and move the two duplicates of each black line slightly apart. This makes a set H of $O(\sqrt{r/8})$ axis-parallel lines such that each bounding box in B intersects at least one line from H.

Then the total number of intersections between the boxes in B and the lines in H is $\Omega(r)$, implying that there is a line in H with stabbing number $\Omega(\sqrt{r/8})$. $\square$

5 Experimental results

In the previous sections we studied the complexity of finding an optimal rectilinear r-partition of a given point set. For arbitrary r the problem is NP-hard, and for constant r the exact algorithm was polynomial but still very slow. Hence, we now turn our attention to heuristics. In the initial experiments on which we report below, the focus is on comparing the various heuristics and investigating the stabbing numbers they achieve as a function of r, for fixed n.

Data sets. We tested our heuristics on four types of point sets–see Fig. 4. The *Uniform* data set picks the points uniformly at random from the unit square. For the *Dense* data set we take a *Uniform* data set and square all y-coordinates, so the density increases near the bottom. For the *Line Clusters* data set we first generated a few line segments, whose endpoints are chosen uniformly at random in the unit square. To generate a point in P, we pick one of the line segments randomly, select a position along the line segment at random and add some Gaussian noise. The *Point Clusters* data set is similar, except that it clusters around points instead of line segments. All sample sets contain $n = 50,000$ points and the reported stabbing numbers are averages over 20 samples.

Next we describe our heuristics. Let P denote the set of points in $\mathbb{R}^2$ for which we want to find a rectilinear r-partition with low stabbing number.

The windmill kd-tree. A natural heuristic is to use a kd-tree [6]: partition the point set P recursively into equal-sized subsets, alternatingly using vertical and horizontal splitting lines, until the number of points in each region drops below $2n/r$. For each region R of the kd-tree subdivision, put the pair $(R \cap P, b_R)$ into

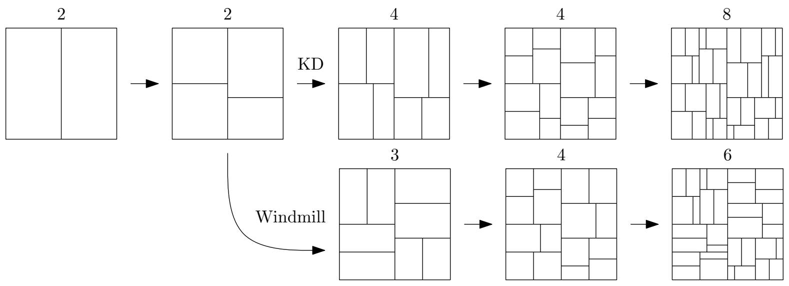

Fig. 5. A kd-tree, a windmill kd-tree, and their stabbing numbers.

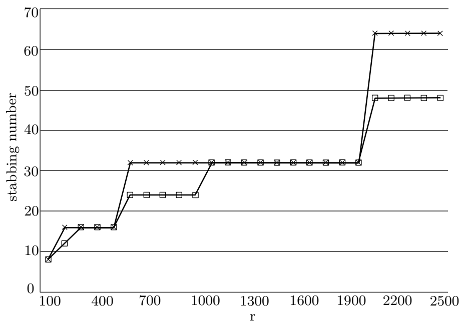

Fig. 6. Comparison of the kd-tree and the windmill kd-tree.

the rectilinear r-partition, where b_R is the bounding box of $R \cap P$. Note that this method *runs in $O(\log n)$ time*, and gives a stabbing number $O(\sqrt{r})$, which is worst-case optimal. The *windmill kd-tree* is a version of kd-tree in which for two of the four nodes of depth 2 the splitting line has the same orientation as the splitting line at their parents. This is done in such a way that the subdivision induced by the nodes at level 2 has stabbing number 3 rather than 4–see Fig. 5. It turns out that the windmill kd-tree is always at least as good as the regular kd-tree, and often performs significantly better. The results for the Random Data set are shown in Fig. 6 for r ranging from 100 to 2,500 with step size 100. The figure shows that, depending on the value of r, the stabbing number of the kd-tree and the windmill kd-tree are either the same, or the windmill has 25% lower stabbing number. The switch between these two cases occurs exactly at the values of r where the depth switches from even to odd (or vice versa), which is as expected when looking at Fig. 5. In the remainder we only compare the windmill kd-tree to the other methods, and ignore the regular kd-tree.

The greedy method. We first compute a set $\mathcal{B}$ of candidate boxes such that $n/2r \leqslant |b_i \cap P| \leqslant 2n/r$ for each box $b_i \in \mathcal{B}$. Each box $b_i \in \mathcal{B}$ has a certain *cost* associated to it. We then take the cheapest box $b_i \in \mathcal{B}$, put the pair $(b_i, P \cap b_i)$ into the rectilinear r-partition, and remove the points in b_i from P . Finally,

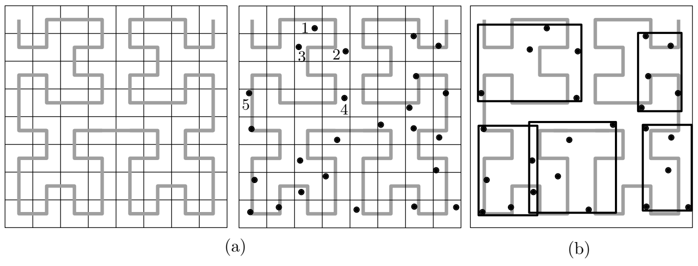

Fig. 7. A Hilbert curve and its use to generate a rectilinear r-partition.

boxes that now contain too few points are discarded, the costs of the remaining boxes are updated, and the process is repeated. The method ends when the number of points drops below $2n/r$; these points are then put into a single box (which we allow to contain less than $n/2r$ points if needed).

This method can be implemented in various ways, depending on how the set $\mathcal{B}$ and the cost of a box are defined. In our implementation we took m vertical lines with (roughly) $n/(m-1)$ points in between any two consecutive lines, and m horizontal lines in a similar manner. $\mathcal{B}$ then consists of all $O(m^4)$ boxes that can be constructed by taking two vertical and two horizontal lines from these lines. In our experiments we used $m = 50$, because this was the largest value that gave reasonable computation times. The cost of a box b_i is defined as follows. We say that a point $p \in P$ is *in conflict with* b_i if $p \notin b_i$ and the horizontal or the vertical line through p intersects b_i. Let C_i be the set of points in conflict with b_i. Then the cost of b_i is $|C_i|/|b_i \cap P|$. The idea is that we prefer boxes containing many points and in conflict with few points.

The Hilbert curve. A commonly used approach to construct R-trees is to use a space-filling curve such as a Hilbert curve [10]—see Fig. 7(a). We can also use a Hilbert curve to compute a rectilinear r-partition: first, sort the given points according to their position on the Hilbert curve, and then generate the subsets in the rectilinear r-partition by taking consecutive subsets along the Hilbert curve. Since the lowest stabbing number is usually achieved by using as few rectangles as possible, we do this in a greedy manner: put the first $2n/r$ points in the first subset, the next $2n/r$ rectangles in the second subset, and so on—see Fig. 7(b).

K-Means. The final method we tested was to compute r clusters using K-means—in particular, we used K-Means++ [3]—and then take the clusters as the subsets in the rectilinear r-partition. Some of the resulting clusters may contain too many or too few points. We solved this by shifting points into neighboring clusters.

5.1 Results of the comparisons

Figs. 8.a-d shows the results of our experiments. The clear conclusion is that the windmill kd-tree outperforms all other methods on all data sets. The Hilbert-curve approach always comes in second, except for the *Dense* data set. Note

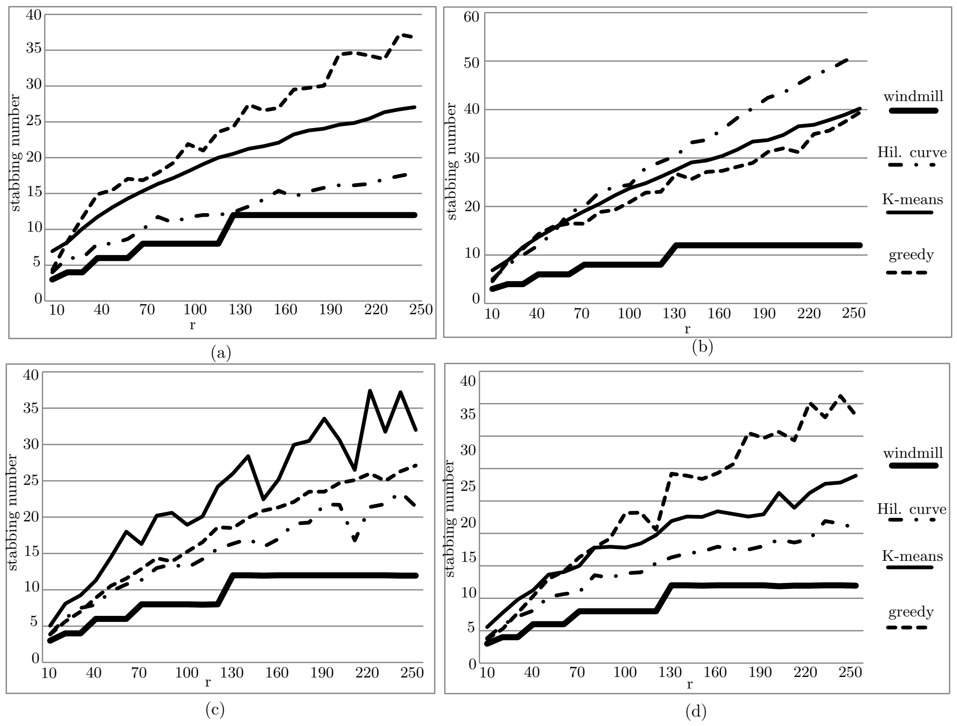

Fig. 8. The results of the comparison of methods on (a)uniform (b)dense (c)line clusters (d)point cluster point sets

that the windmill and the greedy method give the same results for the *Uniform* data set and the *Dense* data set—which is easily explained, since the rectilinear r-partition computed by these methods only depends on the ranks *(their position in the sorted order)* of the coordinates, and not on their actual values—while the other two methods perform worse on the *Dense* data set: apparently they do not adapt well to changing density. The windmill and the Hilbert-curve method not only gave the best results, they were also the fastest. Indeed, both methods could easily deal with large data sets. (On inputs with $n = 10,000,000$ and $r = 500$ they only took a few minutes.)

6 Conclusion

We studied the problem of finding optimal rectilinear r-partitions of point sets. On the theoretical side, we proved that the problem is NP-hard when r is part of the input, although can be solved in polynomial time for constant r. The algorithm for constant r is still unpractically slow, however, so it would be interesting to come up with faster exact algorithms (or perhaps a practically efficient PTAS).

We also tested a few heuristics and concluded that our so-called windmill kd-tree is the best method. This immediately leads to the question whether the windmill approach could also lead to R-trees that are practically efficient. This is, in fact, unclear. What we have tried to optimize is the maximum stabbing number of any axis-parallel line. When querying with a rectangular region, however, we are interested in the number of regions intersected by the boundary of the region. First of all, the boundary does not consist of full lines, but of line segments that in practice are possibly small compared to the data set. Secondly, the boundary of the rectangle consists of horizontal and vertical segments. Now, what the windmill does (as compared to a regular kd-tree) is to balance the horizontal and vertical stabbing number, so that the maximum is minimized. The sum of the horizontal and vertical stabbing number in the subdivision does not change, however. So it might be that the windmill approach is good to minimize the worst-case query time for long and skinny queries. This would require further investigation. It would also be interesting to find rectilinear r-partitions whose (maximum or average) stabbing number is optimal with respect to a given set of query boxes, *or try to minimize the full partition tree, instead of just one level of partition tree, even it seems to be more challenging.*

References

1. Agarwal, P.K., Erickson, J.: Geometric Range Searching and its Relatives. In: B. Chazelle, J. Goodman, and R. Pollack (Eds.), Advances in Discrete and Computational Geometry, 1–56 (1998)
2. Ahuja, P.K., Magnanti, T.L., Orlin, J.B.: Network Flows: Theory, Algorithms, and Applications. Prentice Hall (1993)
3. Arthur, D., Vassilvitskii, S.: K-means++: the Advantages of Careful Seeding. Proc. of the 18th annual ACM-SIAM Sym. of Desc. Alg., 1027–1035 (2007)
4. Chazelle, B., Welzl, E.: Quasi-optimal Range Searching in Spaces of Finite VC-dimension. Arch. Rat. Mech. Anal. 4, 467–490 (1989)
5. Ahuja, P.K., Magnanti, T.L., Orlin, J.B.: Introduction to Algorithms (2nd edition). MIT Press and McGraw-Hill (2001)
6. M. de Berg, O. Cheong, M. van Kreveld, and M. Overmars. Computational Geometry: Algorithms and Applications (3rd edition). Springer-Verlag, 2008.
7. Dijkstra, E.W., Feijen, W.H.J., Sterringa, J.: A Method of Programming. Addison-Wesley, (1988)
8. Fekete, S.P., Lübbecke, M.E., Meijer, H.: Minimizing the Stabbing Number of Matchings, Trees, and Triangulations. Discr. Comput. Geom. 40, 595–621 (2008)
9. Garey, M.R., Johnson, D.S.: Computers and Interactibility: A Guide to the Theory of NP-Completness. W.H. Freeman and Co. (1979)
10. Haverkort, H., van Walderveen, F.: Four-dimensional Hilbert Curves for R-trees. In Proc. Workshop on Algorithms Engineering and Experiments (ALANEX) (2009)
11. Manolopoulos, Y., Nanopoulos, A., Theodoridis, Y., Papadopoulos, A.: R-trees: Theory and Applications. Series in Adv. Inf. and Knowledge Processing, Springer, (2005)
12. Matoušek, J., Tarantello, G.: Efficient Partition Trees. Discr. Comput. Geom. 8, 315–334 (1992)

A Omitted proofs

Recall that a barrier gadget is a $5h \times 5h$ grid for $h = \sqrt{2n/r} = 12$. A convenient way to think about the construction of the gadget is to take a tiny square, partition it into 5×5 subsquares, and placing $2n/r = 144$ points in each of the subsquares. To place 144 points in each cell, we can put 5×11 extra vertical and horizontal lines in barrier gadget such that each subsquare is divided to 144 subcells and then put a point in each of these subcells. See Figure 9 for an illustration.

Lemma 1 *A barrier gadget G can be covered by a set of 25 boxes with stabbing number 5. Moreover, for any rectilinear r-partition $\Psi(S)$ of stabbing number 5, the restriction $\Psi(S \downarrow G)$ has vertical as well as horizontal stabbing number 5.*

Proof. The first part of the lemma is easy: since we can put up to $h^2 \leqslant 2n/r$ points in a box, we can cover all the points from G using 25 square boxes each containing a subgrid of h^2 points.

To prove the second part of the lemma, consider a rectilinear r-partition $\Psi(S)$ of stabbing number 5. Consider the set B containing the boxes in $\Psi(S \downarrow G)$. Each box $b_i \in B$ contains at least one point from G, and together they contain at least $25 \cdot h^2$ points from G. Let L_V be a set of $5(h-1)+6$ vertical lines and L_H be a set of $5(h-1)+6$ horizontal lines that make the grid for the barrier gadget. The horizontal and vertical lines together separate all the points in G—see Fig. 9. Let $L = L_V \cup L_H$. Define $\lambda(b_i)$ to be the number of lines from L that intersect b_i, and define

$$\lambda^*(b_i) := \frac{\lambda(b_i)}{|S_i \cap G|}.$$

For example, if b_i is a box whose associated set $S_i \cap G$ is a $h \times h$ subgrid, then $\lambda(b_i) = 2h - 2$ and $\lambda^*(b_i) = (2h-2)/h^2$.

It is not difficult to verify that $(2h-2)/h^2$ is the minimum possible cost of any box $b_i \in B$. Indeed, consider an $a \times b$ box b_i. Then $|S_i \cap G| \leqslant a \cdot b$. Let us consider the case when $|S_i \cap G| = a \cdot b$ which gives the maximum value for all the $a \times b$ boxes. We have $\lambda(b_i) = a + b - 2$ and the minimum value for $\lambda(b_i)$ is when we choose a and b such that $|b - a| \leqslant 1$. Thus for a $a \times b$ box, when $|b - a| \leqslant 1$ we have the maximum value for $\lambda^*(b_i)$. If we consider all the boxes b_i in which $|b - a| \leqslant 1$, for larger values of a and b the value of $\lambda^*(b_i)$ would be smaller. This is because by increasing a and b the growth of h^2 is more than $2h - 2$. Then the minimum value is when we have the largest box.

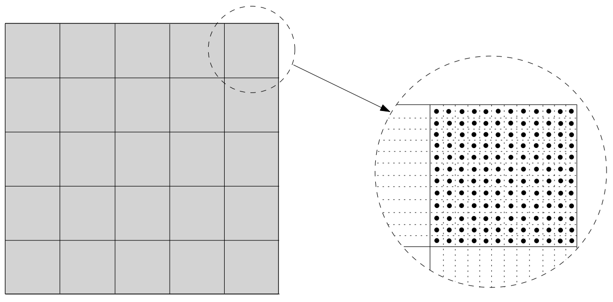

Fig. 9. One of the subsquares of the barrier gadget with 144 points in it.

Thus the total cost of all boxes is

$$\sum_{b_i \in B} \lambda(b_i) \;=\; \sum_{b_i \in B} \left\{ \lambda^*(b_i) \cdot |S_i \cap G| \right\} \;\geqslant\; \frac{2h-2}{h^2} \cdot \sum_{b_i} |S_i \cap G| \;\geqslant\; \frac{2h-2}{h^2} \cdot 25 \cdot h^2 \;=\; 50h - 5.$$

Define $\lambda(\ell)$, for a line $\ell \in L$, as the number of boxes in $\Psi(S \downarrow G)$ that intersect ℓ. Observe that

$$\sum_{\ell \in L} \lambda(\ell) = \sum_{b_i \in B} \lambda(b_i).$$

Now assume for a contradiction that, say, the vertical stabbing number of $\Psi(S \downarrow G)$ is 4 or less. Then

$$\sum_{\ell \in L_H} \lambda(\ell) \;=\; \sum_{\ell \in L} \lambda(\ell) - \sum_{\ell \in L_V} \lambda(\ell) \;\geqslant\; 50h - 5 - 4 \cdot |L_V| \;=\; 30h - 9,$$

This implies that $\sum_{\ell \in L_H} \lambda(\ell) \geqslant 30h - 9$, and for $h \geqslant 12$ this implies that the average number of boxes intersected by each horizontal line is greater than 5, and the lemma holds. □

Theorem 1 Optimal Rectilinear r-Partition *is* NP*-complete for* $k = 5$.
Proof. We can verify in polynomial time whether for a given set of boxes B we can make a rectilinear r-partition with stabbing number of 5 or not, so Optimal Rectilinear r-Partition is in NP.

To prove that Optimal Rectilinear r-Partition is NP-hard, take an instance of 3-SAT with a set $\mathcal{C}$ of m clauses defined over the variables $x_1, \ldots, x_n$. Apply the reduction described above (in the main text) to obtain a set of n points forming an instance of Optimal Rectilinear r-Partition. As we described above we can set the values of k and r so that the relations for making a barrier gadget holds.

Suppose S has a rectilinear r-partition with stabbing number 5. Based on the construction this is only possible if for each clause gadget, at least 4 sets of n/r points (one set of $4n/r$ points) have paired horizontally. When this set of points is placed in the right slab of a variable we set its value to false and when it is placed in the left slab of a variable we set its value to true. Since in each clause there exists 4 sets of n/r points which are paired horizontally. We conclude in each clause there is a variable (or its negation) which is true. Thus, $\mathcal{C}$ is satisfiable.

Now consider a truth assignment to the variables that satisfies $\mathcal{C}$. A rectilinear r-partition for S with stabbing number of 5 can be obtained as follows. For each barrier gadget it has already been shown that how we can make a rectilinear r-partition with stabbing number 5. Based on the values of variables we make the rectilinear r-partition partitions for every variable gadget as shown in Figure 1. For each variable gadget, we pair the sets of points in its slab with stabbing number 2 horizontally and its slab with slabbing number 4 vertically. Since every clause has a true variable, and based on the descriptions for variable and barrier gadgets the vertical and horizontal stabbing numbers of the rectilinear r-partition made is 5. □

Lemma 2. *The modified version of the algorithm in Section 3 reports a rectilinear r-partition with stabbing number at most twice the optimal stabbing number.*

Proof. Let $\Psi := \{(S_1, b_1), \ldots, (S_t, b_t)\}$ be an optimal rectilinear r-partition for S, and let OPT denote the stabbing number of Ψ. Expand every b_j in all directions until each facet of b_j is contained in a hyperplane from H. Let $\overline{b}_j$ denote the expanded box, and let $\overline{\Psi} := \{(S_1, \overline{b}_1), \ldots, (S_t, \overline{b}_t)\}$. The set $\{\overline{b}_1, \ldots, \overline{b}_t\}$ is one of the subsets B considered in Step 2, and it induces a valid solution. Hence, the stabbing number of the reported partition is at most the stabbing number of $\overline{\Psi}$.

Now consider any axis-parallel hyperplane h. Assume without loss of generality that h is orthogonal to the x_1-axis and that h lies in between hyperplanes $h_i, h_{i+1} \in H$. Let $\overline{b}_j$ be a box intersecting h. Note that b_j must intersect h_i or h_{i+1} (or both), otherwise b_j contains too few points. Hence, the h intersects at most $2 \cdot \textsc{opt}$ boxes $\overline{b}_j$. $\square$

Theorem 5. *Let d be a constant, and assume $d \cdot 2^{d+2} \leqslant r \leqslant 2d^{d/2} \cdot \sqrt{n}$. Then there is a set S of n points in $\mathbb{R}^d$ whose optimal rectilinear r-partition has stabbing number 2, while any rectilinear r-partition with disjoint bounding boxes has stabbing number $\Omega(r^{1-1/d})$.*

Proof. Let $\mathcal{G}$ be a $(r/2d^d)^{1/d} \times \ldots \times (r/2d^d)^{1/d}$ grid in $\mathbb{R}^d$. (We assume for simplicity that $(r/2d^d)^{1/d}$ is an integer; note that since $d \cdot 2^{d+2} \leqslant r$ we have $(r/2d^d)^{1/d} \geqslant 2$.) We put each grid point in S. We call these points *black points*, and we call the hyperplanes forming the grid $\mathcal{G}$ *black hyperplanes.* Note that there are $r/2d^d$ black points. Fig. 3 shows an example for $d = 2$ with $r = 128$. Next we refine the grid using $d((r/2d^d)^{1/d} - 1)$ additional axis-parallel *grey hyperplanes.* At each of the new grid points that is not fully defined by gray hyperplanes—the grey dots in the figure—we put a tiny cluster of $2n/r$ points, which we also put in S. If the cluster lies on one or more black hyperplanes, then all points from the cluster lie in the intersection of those hyperplanes.

So far we used

$$((2((r/2d^d)^{1/d} - 1))^d - ((r/2d^d)^{1/d} - 1)^d - ((r/2d^d)^{1/d})) \cdot 2n/r$$

points; in gray dots. Since we have r $\leqslant 2d^{d/2} \cdot \sqrt{n}$ it is easy to show that this number is less than $(r/2 - 1) \cdot 2n/r$. Moreover $(r/2d^d)$ (the number of black points) is less than $2n/r$. Thus the total number of points we have put so far is less than n. The remaining points can be placed far enough from the construction, not influencing the coming argument.) Next, we rotate the whole construction slightly *around a line not parallel to axis lines*, so that no two points have the same coordinate in any dimension. This rotated set is our final point set S.

To obtain a rectilinear r-partition with stabbing number 2, we make each of the clusters into a separate subset S_i, and we put the black points into one separate subset; the latter is allowed since $r/d^2 \leqslant 2n/r$. (If $r < n/2r$ we can use some of the remaining points or the points of gray dots to fill up the subset.) If the clusters are small enough, then the rotation we have applied to the point set guarantees that no axis-parallel hyperplane can intersect two clusters at the same time. Hence, the stabbing number of this rectilinear r-partition is 2.

We claim that any disjoint rectilinear r-partition for S has stabbing number $\Omega(r^{1-1/d})$. To see this, observe that no subset S_i in a disjoint rectilinear r-partition can contain two black points. Indeed, the bounding box of any two black points contains at least one full cluster and, hence, together with the black points would be too many points. We conclude that each black point is assigned to a different bounding box. Let B be the collection of these bounding boxes. Now consider a set H of $O(r^{1/d})$ axis-parallel hyperplanes such that each bounding box in B intersects at least one hyperplane from H. (Such a set can be found by duplicating each of the black hyperplanes, and moving the two duplicates of each black hyperplane slightly apart.) Then the total number of intersections between the boxes in B and the hyperplanes in H is r, which implies that there is a hyperplane in H with stabbing number $\Omega(r^{1-1/d})$.