Best Fitting Rectangles

Manuel Abellanas[*] Ferran Hurtado[**] Christian Icking[**]
Lihong Ma[**] Belén Palop[***] Pedro A. Ramos[****]

Technical Report 322, Dept. of Computer Science, FernUniversität Hagen
December 2004

Abstract

We solve an interesting optimization problem motivated by tolerancing metrology, paper position sensing, and facility location: What rectangle fits best a given set of points? This problem has relations to the minimum width annulus problem and also to the largest empty rectangle problem for both of which $O(n \log n)$ time solutions are not known. Nevertheless, for our problem there is a linear time algorithm for arbitrary axis-parallel rectangles and a $O(n \log n)$ solution if the aspect ratio of the rectangles is prescribed.

1 Introduction

The task of finding a shape that fits nicely to a set of n points in the plane has many applications. In computational metrology [18, 19, 31], for example, a common task is to test the circularity of an object, an this is usually done by testing a sufficiently large set of sample points on the surface of the object and determining a circle that fits best to these points; this immediately leads to the *minimum width annulus problem*: For a set of n points in the plane compute two concentric circles such that all points are contained between them and the difference of the two radii is minimized. Rivlin [27] shows that the optimal solution contains at least two of the points on the inner as well as on the outer circle and that it can also degenerate to a slab of two parallel lines; compare also Smid and Janardan [29].

Algorithms for the minimum width annulus problem that run in time $O(n^2)$ have been proposed by Wainstein [30], Ebara et al. [17], and Roy and Zhang [28]. García-López

[*] Departamento de Matemática Aplicada, Universidad Politécnica de Madrid, Spain
[**] Departament de Matemàtica Aplicada II, Universitat Politècnica de Catalunya, Barcelona, Spain
[***] Praktische Informatik VI, FernUniversität Hagen, Germany
[****] Departamento de Informática, Universidad de Valladolid, Spain
[*****] Departamento de Matemáticas, Universidad de Alcalá, Spain

et al. [20] characterize the local minimum for this problem in any dimensions and, for two dimensions and a given circular order of the points, show how to find the optimum that is consistent with this particular order in time $O(n \log n)$. By applying sophisticated techniques like parametric search, see Megiddo [23], the running time to solve the general problem can be improved, at least in theory, to $O(n^{\frac{3}{2}+\varepsilon})$ for any $\varepsilon > 0$, see Aggarwal et al. [1, 2, 3].

A more practical approach makes use of the fact that the center of the optimal annulus must be at an intersection of an edge of the Voronoi diagram and an edge of the farthest-point Voronoi diagram of the point set (except for the special degenerate case mentioned above than can be treated separately, see Houle and Toussaint [22]), and all such intersections can be computed by the general superposition method of Guibas and Seidel [21]. If k is the number of these intersections, one obtains a relatively simple $O(n \log n + k)$ algorithm which is even more interesting by the fact that k, although in $\Theta(n^2)$ in the worst case, is expected to be in $O(n)$ in random point sets.

For a variation of the minimum width annulus problem in which a radius r is given, Duncan et al. [16] as well as Bose et al. [12] give algorithms to compute the optimal annulus whose outer, inner, or median radius equals r in time $O(n \log n)$. For the largest *empty* annulus see Díaz-Báñez et. al. [13].

Other variations of the problem concern the shape of the object. Offset polygon problems, for example, are treated by Barequet et. al., see [6, 7].

In this paper, we propose a solution for rectangular shapes, either arbitrary rectangles or ones with given aspect ratio. Such a best fitting rectangle, in many cases, will contain a quite large empty rectangle. So at first sight, the best fitting rectangle problem has much in common with the *largest empty rectangle problem*: For a set of n points in the plane compute the rectangle with maximum area that does not contain any of the points in its interior and which is contained in the bounding box of the points.

The largest empty rectangle problem has attracted a lot of attention in the past years, for example see [5, 10, 25]. For the axis-parallel version the best known solutions are by Aggarwal and Suri [4] for the worst case in time $O(n \log^2 n)$ and by Orlowski [26] whith an $O(n \log n)$ expected time. For the largest empty rectangle inside a polygon see Daniels et. al. [11]. Dropping the restriction of axis-parallelism and allowing arbitrarily oriented rectangles makes the problem more complicated, but two $O(n^3)$ worst case time solutions which use similar methods were independently given by Mukhopadhyay and Rao [24] and by Chaudhuri and Nandy [9].

More applications for this kind of problems can be found in paper position sensing, see [8], and facitily location, for example et. al. [14, 15].

2 Notations

We are given a set P of n point sites in the plane. Our task is to determine the rectangle which is, in some sense, closest to all of them.

As usual, the distance between a point and an extended object means the distance between the point and the closest point on the object, so the distance between a point p and a rectangle R is

$$d(p, R) = \min_{q \in R} d(p, q) \,.$$

Here, $d(p, q)$ denotes the distance in the underlying metric. Note that by rectangle we mean the boundary of the rectangle, not the interior. So a point in the interior of a rectangle has a non-zero distance to the rectangle.

For a certain subset, $\mathcal{R}$, of admitted rectangles, we are looking for a best fitting one, i. e., a rectangle such that

$$\max_{p \in P} d(p, R)$$

is minimized over all rectangles R of that kind. In other words, a best fitting rectangle R_{opt} fulfills

$$\max_{p \in P} d(p, R_{opt}) \;=\; \inf_{R \in \mathcal{R}} \max_{p \in P} d(p, R) \,. \tag{1}$$

Different kinds of admitted rectangles, $\mathcal{R}$, and different metrics generate different problems to be solved. If the set $\mathcal{R}$ is closed then we may write min instead of inf in (1), and a best fitting rectangle $R_{opt} \in \mathcal{R}$ is guaranteed to exist. This is clearly the case for the problems considered in this paper.

An environment of a rectangle is called a frame. More precisely, the set of points whose distance to a rectangle R is less or equal to ε is called the *ε-frame* of R, for short F_R^ε. The two closed boundaries of F_R^ε are called the *outer and inner ε-offsets* of R. The ε-frame of R is also the Minkowski sum of R and the unit circle of the underlying metric, scaled by ε.

In this paper we concentrate on axis-parallel rectangles with or without prescribed aspect ratio, and we will use the L_∞-distance as underlying metric, which has axis-parallel squares as unit circles. Therefore, the outer and inner offsets of a rectangle are also rectangles, see Figure 1.

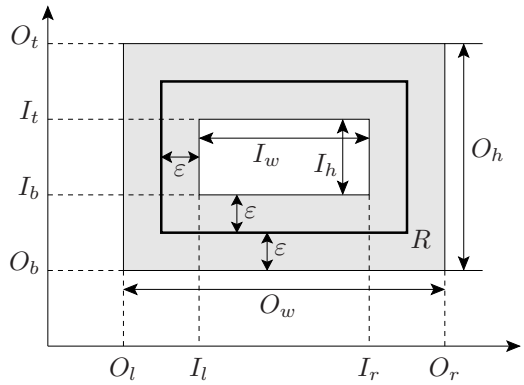

Figure 1: A rectangle, R, and its ε-frame, F_R^ε, under the L_∞-metric.

The *best fitting rectangle problem* is equivalent to computing a narrowest covering ε-frame, i. e., the frame with smallest possible ε that covers all sites. Remark that in many

cases the best fitting rectangle is not unique, and our algorithms will find a smallest one among all best fitting, as we will see.

The following notations will be used. For the Y-coordinates of the horizontal offset edges of a certain frame we say O_t, O_b, I_t, and I_b to the outer and inner top and bottom edges, and for the X-coordinates of the vertical offset edges we say O_l, O_r, I_l, and I_r to the outer and inner left and right edges. The heights and widths of the outer and inner frame are denoted by $O_h = O_t - O_b$, $O_w = O_r - O_l$, $I_h = I_t - I_b$, and $I_w = I_r - I_l$, respectively, see Figure 1.

With $x_{\min}$, $x_{\max}$, $y_{\min}$, and $y_{\max}$ we denote the coordinates of the bounding box, B, of the point set P, and with $H = y_{\max} - y_{\min}$ and $W = x_{\max} - x_{\min}$ we mean its height resp. width.

3 Arbitrary aspect ratio

If we do not prescribe a certain aspect ratio of the rectangle then computing a best fitting rectangle is an easy problem.

Lemma 1 *Let R be a rectangle whose ε-frame covers P and let B the bounding box of P. Then we have for all $p \in P$*

$$d(p, B) \leq 2\varepsilon \ .$$

Proof. Let $p \in P$ and let r be the point of R closest to p (one of them if there are several); we know $d(p, r) \leq \varepsilon$.

The (horizontal or vertical) line through p and r intersects B in two points. It is clear that at least one of the two is not farer than ε away from r because otherwise one of the sites on B would not be covered by the ε-frame of R, thus $d(r, B) \leq \varepsilon$.

We can combine the two inequalities and use the triangle inequality to obtain

$$d(p, B) \leq d(p, r) + d(r, B) \leq 2\varepsilon \ ,$$

which is our claim. □

Now an optimal solution can be obtained as follows.

- Compute B, the bounding box of P, this will turn out to be the outer offset of an optimal ε-frame.
- Compute $\max_{p \in P} d(p, B)$; call this 2ε.
- Let R be the inner ε-offset of B; this is a best fitting rectangle.

To prove the correctness of the algorithm, which clearly runs in time $O(n)$, it suffices to say that F_R^ε covers all sites and that there is no covering frame of a smaller width, by Lemma 1.

4 Given aspect ratio

Now the aspect ratio, $a = \dfrac{height}{width}$, of the considered rectangles is given. The problem is more complicated as in the previous section because the bounding box is no longer such a direct key to the solution. Nevertheless, the following lemma says that we will have some points on the boundary of an optimal ε-frame.

Lemma 2 *The inner and the outer ε-offsets of any best fitting rectangle with given aspect ratio, a, contain a point of P.*

Proof. Let F_R^ε be a covering frame.

(i)  If there is no point of P on the offset edges of F_R^ε, then by continuity there is an $\varepsilon' < \varepsilon$ such that for the same R its frame $F_R^{\varepsilon'}$ is also covering, thus ε is not optimal.

(ii)  If there are points of P only on the outer offset edges of F_R^ε, but not on the inner offset edges, then we can scale R about its center point by a factor slightly bigger than 1 to a rectangle R', such that for the same ε the frame $F_{R'}^\varepsilon$ is also covering but contains no points on its offset edges, and (i) applies.

(iii)  If there are points of P only on the inner offset edges of F_R^ε, but not on the outer offset edges, then we can scale R about its center point by a factor slightly smaller than 1 to a rectangle R'', such that for the same ε the frame $F_{R''}^\varepsilon$ is also covering but contains no points on its offset edges, and (i) applies again. □

The following lemma shows some relations between the widths, O_w and I_w, and the heights, O_h and I_h, of outer and inner offsets, recall also Figure 1.

Lemma 3 *Let R be a rectangle with aspect ratio a, and let $\varepsilon > 0$. For the dimensions of the ε-frame, F_R^ε, we have the following equations depending on the outer height, O_h.*

$$O_w = \frac{O_h}{a} + 2\varepsilon(1 - \frac{1}{a}) \tag{2}$$

$$I_w = \frac{O_h}{a} - 2\varepsilon(1 + \frac{1}{a}) \tag{3}$$

$$I_h = O_h - 4\varepsilon \tag{4}$$

Proof. We have

$$a = \frac{O_h - 2\varepsilon}{O_w - 2\varepsilon} = \frac{O_h - 2\varepsilon}{I_w + 2\varepsilon}\ .$$

Solving these equations for O_w resp. I_w result in (2) resp. (3), while (4) is true by definition. □

Now we can improve on Lemma 2 and describe in a precise manner the essential cases for points on the offset edges. Essential in this case means that we are not interested in finding *all* best fitting rectangles with equal ε but only *one*, the smallest one. Therefore, in the following lemma we only claim the existence of such cases; there may be other best fitting rectangles with the same ε whose offsets touch less points of P (but at least two, of course, by Lemma 2). For brevity, we say an edge is *touching* if it is touching a point of P. See the end of Section 2 for the meanings of O_t, O_b, etc.

Lemma 4 *There is always a best fitting rectangle with given aspect ratio, a, that corresponds to one of the following three main cases, possibly after a rotation of P by $\pm 90°$ or $180°$, see Figure 2:*

Case 1 *O_t, O_b, and I_t are determined by three points of P.*

Case 2 *O_t, O_b, I_l, and I_r are determined by four points of P.*

Case 3 *O_t, O_l, O_b, and I_l are determined by three or four points of P. Only if a point of P lies on a corner of the bounding box of P then three points can determine Case 3, otherwise they are four.*

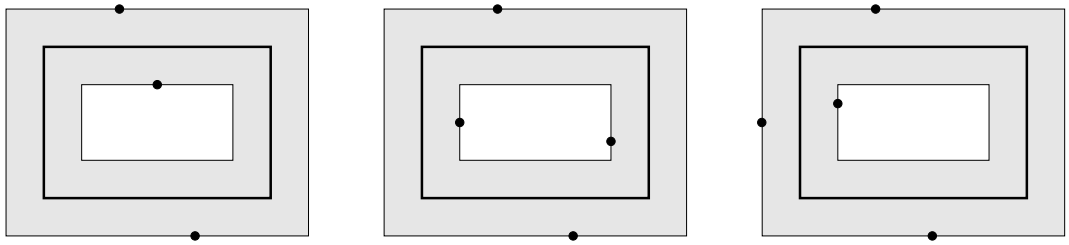

Figure 2: The three main cases of best fitting rectangles for a given aspect ratio.

Proof. Let F_R^ε be an optimal covering frame. From Lemma 2 we know that there are points of P on the outer and inner offset. W.l.o.g. we assume that the outer top edge is touching, this is denoted by  , all other cases are obtained by rotation of P by $\pm 90°$ or $180°$. Now there are essentially three different positions for a point on the inner offset which must exist by Lemma 2, it can touch the inner top, the inner left (which is symmetric to the inner right), and the inner bottom edges.

(i)  The inner top edge is touching. We describe a continuous shrinking process which will lead to one of the given cases. Parameter ε will always be constant, and the outer and inner top edges keep their supporting lines.

The parameter for this shrinking process is the outer height, O_h, which is continuously decreasing. The other quantities, O_w, I_w, and I_h are also continuously decreasing with O_h, see (2), (3), and (4). The shrinking is continued only as long as all points of P are covered, of course.

Even if the outer left edge contains a point, it is still possible to shrink further with fixed O_l and I_l as long as the outer right edge contains no point, and vice versa. Therefore the shrinking can stop only if the bottom edge becomes touching,  , this is Case 1, or if the outer left *and* right edges touch points,  , this is Case 3 rotated by 90°, see Figure 2.

(ii)  The inner left edge is touching, but the inner top edge is not. At least one of the outer left or the inner right edges must also touch, otherwise we can shift the frame slightly to the right, the inner offset would touch no point, and ε would not be optimal by Lemma 2; so we distinguish two sub-cases.

(ii.a)  The outer left edge is also touching. After a rotation by 90° this is treated by (i).

(ii.b)  The inner right edge is also touching, but the outer left and right edges are not. Here, if the outer bottom edge is not also touching, then we can shift the frame slightly upwards, the outer offset would touch no point at all, and ε would not be optimal by Lemma 2. The only remaining possibility is that the outer bottom edge is also touching,  , and we are in Case 2.

(iii)  The inner bottom edge is touching, but the inner top, left, and right edges are not. If now the outer bottom edge is not touching then we can shift the frame slightly to the top, the inner offset would touch no point, and ε would not be optimal by Lemma 2 once again. Thus, the outer bottom edge is touching,  , and we are back in (i) after a rotation of 180°. □

The above lemma is the base for our algorithm which will check for all occurences of these cases and determine the covering frame from one of the three cases with the smallest ε.

The following lemma makes sure that a reasonable covering frame always exists, namely a frame whose inner offset is empty. This can be used as an initial setting in an algorithm. Recall that $x_{\min}$, $x_{\max}$, etc., denote the coordinates of the bounding box of P, and H and W its height resp. width.

Lemma 5 *There is always a covering frame, F_R^ε, for the point set P, either with touching outer top and bottom edges and*

$$\varepsilon = \frac{H}{4} \min\left(1, \frac{2}{1+a}\right)$$

or with touching outer left and right edges and

$$\varepsilon = \frac{W}{4} \min\left(1, \frac{2}{1+\frac{1}{a}}\right)$$

Proof. We will construct a covering frame whose inner offset is just a line segment, i. e., $I_h = 0$ or $I_w = 0$, depending on whether $a \leq 1$ or $a > 1$. In other words, the frame will be just a rectangle without a hole and we will make sure that the bounding box of P is contained in it.

First let us consider $a \leq 1$ which implies $I_h = 0$ and $\varepsilon = O_h/4$, see the left picture in Figure 3. By (2) we have $O_w = O_h/a + O_h/2(1 - 1/a) = O_h/2(1 + 1/a)$.

Now there are two possibilities: if $W \leq H/2(1 + 1/a)$ then let $O_h = H$, i. e., the outer top and bottom edges are touching. For the width we get $O_w = H/2(1+1/a) \geq W$, which is sufficient to cover P.

Otherwise, i. e., $W > H/2(1 + 1/a)$, let $O_w = W$ and the outer left and right edges are touching, and for the height we get $O_h = 2W/(1 + 1/a) \geq H$, which is sufficiently high.

In both cases we have found a covering frame.

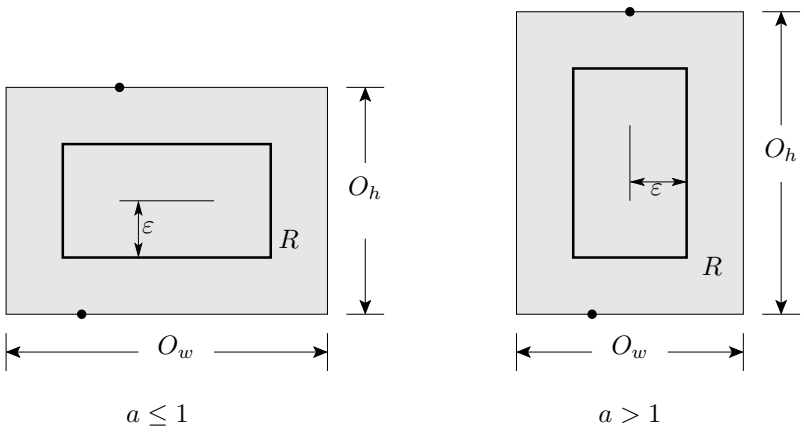

Figure 3: Frames without a hole.

For $a > 1$ which implies $I_w = 0$ and $\varepsilon = O_w/4$ this is similar, see the right picture in Figure 3. By (2) we have $O_h = aO_w - O_w/2(a-1) = O_w/2(1+a)$.

If now $H \leq W/2(1+a)$ then let $O_w = W$ which means that the outer left and right edges are touching, and $O_h = W/2(1+a) \geq H$ is big enough.

Otherwise, i. e., $H > W/2(1+a)$, then let $O_h = H$ and the outer top and bottom edges are touching while $O_w = 2H/(1+a) > W$ is sufficiently large.

If we now recombine the formulae concerning ε in the cases when $O_h = H$ we get the following.

$$\left.\begin{array}{ll} \varepsilon = \dfrac{H}{4} & \text{if } a \leq 1 \\ \varepsilon = \dfrac{H}{2(1+a)} & \text{if } a > 1 \end{array}\right\} \Leftrightarrow \varepsilon = \frac{H}{4} \min\left(1, \frac{2}{1+a}\right)$$

Analogously, for the cases in which $O_w = W$ we obtain $\varepsilon = \dfrac{W}{4} \min\left(1 + \dfrac{2}{1+\frac{1}{a}}\right)$. □

Lemma 5 makes also clear that if we are given the outer height, O_h, of a frame then

$$\varepsilon_{\max} = \frac{O_h}{4} \min\left(1, \frac{2}{1+a}\right)$$

is the maximum reasonable value for ε, and similar for a given width.

On the other hand, it is important to note that even if $O_h = H$ is given and the outer width corresponding to the said maximum ε-value is not big enough to cover all points, at least for $a > 1$ there is still a chance of a covering frame with $O_h = H$ and a smaller ε! This is because a *decreasing* ε with constant O_h implies an *increasing* O_w by (2) if $a > 1$.

So let us now attack each of the three cases of Lemma 4 separately by giving and proving an algorithm. For the simplicity or the description we assume general position, i. e., no two points of P have identical X- or Y-coordinates. Nevertheless the treatment of all special positions is not difficult at all.

Algorithm 1. We perform a sweep from the middle of the Y-interval $(y_{\min}, y_{\max})$ simultaneously to the top and to the bottom. Thereby we check all interesting positions where a covering frame corresponding to **Case 1** rotated by 0° or by 180° may exist.

Input: point set P and aspect ratio a.

$O_t = y_{\max}$, $O_b = y_{\min}$, $O_h = O_t - O_b$,
they are fixed for this case and correspond to the bounding box.

For all points $p = (x, y)$ of P compute $\varepsilon(p) = \frac{1}{2}\min(O_t - y, y - O_b)$.
Sort the points according to decreasing ε-values
    and renumber the points, p_1, p_2, ..., in that order.
Let $\varepsilon_i = \varepsilon(p_i)$ and $\varepsilon_{start} = \frac{O_h}{4}\min\left(1, \frac{2}{a+1}\right)$.
Let T be an empty balanced tree to contain points according to their X-coordinates.
Insert all points p_i with $\varepsilon_i > \varepsilon_{start}$ into T.
For all remaining points $p_i = (x_i, y_i)$ in this order do
    Let $I_w = \frac{O_h}{a} - 2\varepsilon_i\left(1 + \frac{1}{a}\right)$ be the width of the inner offset, see (3).
    Search in T the two subsequent $p_l = (x_l, y_l)$, $p_r = (x_r, y_r) \in T$ with $x_l < x_i < x_r$.
    Let $h_l = \min(x_{\min}, x_i - 2\varepsilon_i, x_r - 2\varepsilon_i - I_w)$
    and $h_g = \max(x_l - 2\varepsilon, x_i - 2\varepsilon_i - I_w, x_{\max} - 4\varepsilon_i - I_w)$.
    If $h_l \le h_g$ then
        we have found a narrower covering frame with
        $O_l = h_l$, $I_l = h_l + 2\varepsilon_i$, $I_r = I_l + I_w$, $O_r = I_r + 2\varepsilon_i$,
        $I_t = O_t - 2\varepsilon_i$, and $I_b = O_b + 2\varepsilon_i$.
    Insert p_i into T.

Output: The last covering frame found, if any.

Lemma 6 Algorithm 1 *finds a narrowest covering frame corresponding to* Case 1*, i. e., with $O_t = y_{\max}$, $O_b = y_{\min}$, and a point of P on the inner top or bottom edge, if such a frame exists. Its running time is in $O(n \log n)$.*

Proof. The algorithm chooses its initial value ε_{start} such that the inner offset is just a line segment, see Lemma 5. All points of P lie between $O_b = y_{\min}$ and $O_t = y_{\max}$, and we partition this area into three horizontal strips: the upper and lower strips are each 2ε high, while the middle strip has height $I_h = O_h - 4\varepsilon$, and the tree T contains at each time the points of the middle strip, i. e., the points $p = (x, y)$ with $O_b + 2\varepsilon < y < O_t - 2\varepsilon$, see Figure 4.

In the For-loop, the current point is called $p_i = (x_i, y_i)$ and at each iteration it is checked whether the point p_i can lie on the inner top or bottom edge of a covering frame with $\varepsilon = \varepsilon_i$. The inner width, I_w, is known to be $\frac{O_h}{a} - 2\varepsilon_i(1 + \frac{1}{a})$, by (3). For a frame with outer left edge at O_l to cover P it is necessary and sufficient to fulfill the following two conditions.

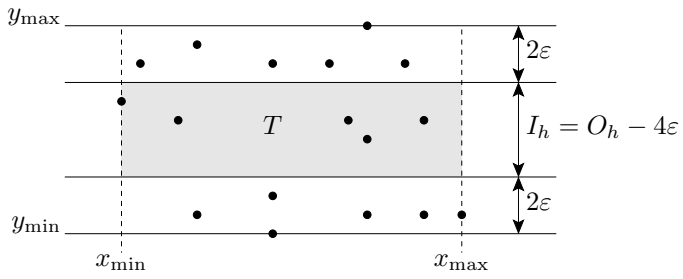

Figure 4: The three horizontal strips for Algorithm 1.

1. The bounding box of P must be contained in the outer offset of the frame, in particular this makes sure that the points of the upper and lower strips are covered:

$$\begin{aligned} O_l &\leq x_{\min} \\ O_l &\geq x_{\max} - 4\varepsilon_i - I_w \end{aligned}$$

2. All points in the middle strip (i. e., in T) must not be contained in the inner offset of the frame, while p_i must lie on the inner upper or lower edge (here we make use of the assumption that all X- and all Y-coordinates are pairwise different). If we call $p_l = (x_l, y_l) \in T$ the predecessor of p_i in X-order and $p_r = (x_r, y_r)$ its successor, this can be expressed as follows:

$$\begin{aligned} O_l &\geq x_l - 2\varepsilon_i \\ O_l &\leq x_i - 2\varepsilon_i \\ O_l &\geq x_i - 2\varepsilon_i - I_w \\ O_l &\leq x_r - 2\varepsilon_i - I_w \end{aligned}$$

The inequalities above are given in a form solved for O_l for a good reason. Namely, the crucial observation is that the frame is covering if and only if a number O_l exists that fulfills all six inequalities above. The algorithm makes use of this in a simple way: the three "$\leq$"-inequalities are combined in h_l and the "$\geq$"-inequalities in h_g. Now it is clear that there is a covering frame for the current ε_i if and only if $h_l \leq h_g$, and we can choose $O_l = h_l$ if it exists.

Parameter ε is decreased step by step, the upper and lower strips become narrower, and more and more points are contained in T. The last covering frame found has the smallest ε, i. e., represents the best fitting rectangle for Case 1, but it can happen that there is no covering frame at all in this case.

The running time results from the sorting and from the $O(n)$ accesses to the balanced tree. □

Algorithm 2. We perform a sweep but this time from left to right. Thereby we check all interesting positions where a covering frame corresponding to **Case 2** may exist.

Input: point set P and aspect ratio a.

$O_t = y_{\max}$, $O_b = y_{\min}$, $O_h = O_t - O_b$,
they are fixed for this case and correspond to the bounding box.

Sort all points $p = (x, y)$ of P according to increasing X-coordinates and renumber the points, p_1, p_2, ..., in that order.
succ(p) returns the immediate successor of point p in the sorted order.

We use two points p_l resp. p_r of P that determine the inner left resp. inner right edge of a frame.
$p_l = p_1$ (i. e., $p_{l_x} = x_{\min}$)
$p_r = p_2$

Let T be a balanced tree, initially empty. It will always contain the points of the current X-interval (p_{l_x}, p_{r_x}), sorted by their Y-coordinates.

while true
$I_w = p_{r_x} - p_{l_x}$
$\varepsilon = \dfrac{O_h - aI_w}{2(1+a)}$, see (3).
$O_l = p_{l_x} - 2\varepsilon$, $O_r = p_{r_x} + 2\varepsilon$, $I_b = O_b + 2\varepsilon$, $I_t = O_t - 2\varepsilon$

(L1) if $x_{\min} < p_{l_x} - 2\varepsilon$ then stop algorithm.

(L2) if T contains at least one point in the Y-interval (I_b, I_t) then
remove p_l from T
$p_l = succ(p_l)$
continue while

(L3) if $p_{r_y} \notin [I_b, I_t]$ then
insert p_r into T
if succ(p_r) exists then $p_r = $ succ(p_r) else stop algorithm
continue while

(L4) if $p_{l_y} \in [I_b, I_t]$ and $x_{\max} \le O_r$ then
this is a covering frame with O_l, $I_l = p_{l_x}$, $I_r = p_{r_x}$, O_r, O_b, I_b, I_t, O_t
remember this frame if its ε is smaller than the best known so far.

(L5) make T empty
$p_l = p_r$
if succ(p_r) exists then $p_r = $ succ(p_r) else stop algorithm
end while

Output: the last frame remembered after (L4), if any.

Lemma 7 Algorithm 2 *finds a narrowest covering frame corresponding to* Case 2*, i. e., with* $O_t = y_{\max}$*,* $O_b = y_{\min}$*, and points of* P *on the inner left and right edges, if such a frame exists. Its running time is in* $O(n \log n)$*.*

Proof. In contrast to Case 1, sorting according to ε seems not to be possible in Case 2. Therefore the algorithm proceeds from left to right and tests all interesting pairs of points, p_l and p_r, on the inner left and right edges. Both points do only advance and never step backwards. Thus the running time is dominated by the sorting and by the $O(n)$ accesses to the balanced tree.

All points of P obviously lie between $O_b = y_{\min}$ and $O_t = y_{\max}$. In X-direction the points are partitioned into three vertical strips: a middle strip of the points between p_{l_x} and p_{r_x}, they are contained in the balanced tree T, and a left and a right strip, see Figure 5.

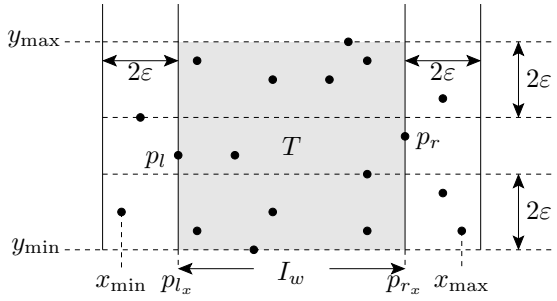

Figure 5: The three vertical strips for Algorithm 2.

In the while-loop, at each iteration the current points p_l and p_r are checked if they can lie on the inner left resp. right edges of a covering frame. The inner width, I_w is just the X-distance between these two points, and ε can then be computed by (3). Thus the frame is completely determined, i. e., the quantities O_l, O_r, I_b, and I_t are known, and it is covering if and only if five conditions are fulfilled.

1. The point with the smallest X-coordinate, $x_{\min}$, must be covered, this is tested at (L1).

   Furthermore, if $x_{\min}$ is not covered then we know the following.

   - If we keep the inner left edge $I_l = p_{l_x}$ and shift the right edge to the right, $I_r \geq p_{r_x}$, then also I_w gets bigger, ε becomes smaller and therefore O_l bigger, so $x_{\min}$ is still not covered.
   - On the other hand, if only the inner left edge moves to the right, then although ε becomes bigger the outer left edge also moves to the right, this can be verified using (3). Therefore, $x_{\min}$ will also not be covered.
   - Thus, for any frame with $I_l \geq p_{l_x}$ and $I_r \geq p_{r_x}$ we know that $x_{\min}$ will never again be covered. It makes no sense to proceed with the algorithm, so we can stop here.

2. The points in the middle strip, i. e., in T, are not contained in the Y-interval (I_b, I_t) which is equivalent to the inner offset of the current frame being empty, this is checked at (L2).

   But if there is at least one such point then there cannot be a covering frame with $I_l = p_{l_x}$ and any bigger $I_r \geq p_{r_x}$, because for any such frame we know that it has a bigger inner width, I_w, and a smaller ε as the current frame. The inner offset of such a frame completely contains the inner offset of the current frame and cannot be empty. Therefore, we can advance p_l without the risk of missing an interesting position. The statement "continue while" means that the algorithm continues at the beginning of the while loop.

3. The point p_r must be contained in the Y-interval $[I_b, I_t]$, i. e., p_r lies on the boundary of the inner frame as required for Case 2, this is checked at (L3).

   But if it is not then we advance p_r while p_l remains the same point.

4. The point p_l must also be contained in the Y-interval $[I_b, I_t]$, i. e., p_l lies on the boundary of the inner frame as required for Case 2, this is checked at (L4).

5. The point with the biggest X-coordinate must be covered, this is also tested at (L4).

   If all five conditions are fulfilled then this frame of Case 2 is covering all points. It has to remembered if ε is smaller than the best known before.

   Furthermore, for each pair (p_l, p_r) that is tested at (L4), i. e., which has passed the tests at (L1), (L2), and (L3), but independently of the outcome of the test at (L4), we know the following.

   - There cannot be a covering frame with $I_l = p_{l_x}$ and any bigger $I_r \geq p_{r_x}$ because for any such frame the current point p_r lies in the interior of the inner offset, this is very similar to the argument at 2.
   - And there also cannot be a covering frame with an inner left edge between p_l and p_r and an inner right edge on or to the right of p_r because such a frame either has a bigger ε and contains p_r inside the inner offset or it has a smaller ε and the inner left edge is contained in the current inner offset and cannot touch a point.

     (In principle, in a very special case there can be such a frame with an ε identical to the current one and a point on an inner corner, this frame will have been treated as Case 1.)

   Therefore, we can advance p_l immediately to p_r, make the tree T empty and continue from this position.

The algorithm comes to an end when a "stop algorithm" statement is reached either after (L1), (L3), or (L5). □

Algorithm 3. We check for **Case 3**, i. e., we look for a covering frame with points of P on the outer top, left, and bottom edges as well as on the inner left.

Input: point set P and aspect ratio a.

$O_t = y_{\max}$, $O_l = x_{\min}$, $O_b = y_{\min}$, $O_h = O_t - O_b$,
they are fixed for this case and correspond to the bounding box.

Let Δ be the isosceles, right-angled triangle whose hypotenuse
is the left edge of the bounding box.
If $a > 1$ then

from Δ we cut off the part which is right to the vertical line
at $X = O_l + \dfrac{O_h}{1+a}$ such that Δ is now a trapezoid.

See the Figure 6 for the two cases $a \leq 1$ and $a > 1$.

Among all points of P that are contained in Δ determine
the point p_l with maximum X-coordinate.

Consider the frame with
$I_l = p_{l_x}$, $\varepsilon = \dfrac{I_l - O_l}{2}$
$I_r = I_l + \dfrac{O_h}{a} - 2\varepsilon_i \left(1 + \dfrac{1}{a}\right)$, see (3).
$O_r = I_r + 2\varepsilon$, $I_t = O_t - 2\varepsilon$, $I_b = O_b + 2\varepsilon$

For all points of P check if they are covered by this frame.

Output: the frame found, if it is covering P.

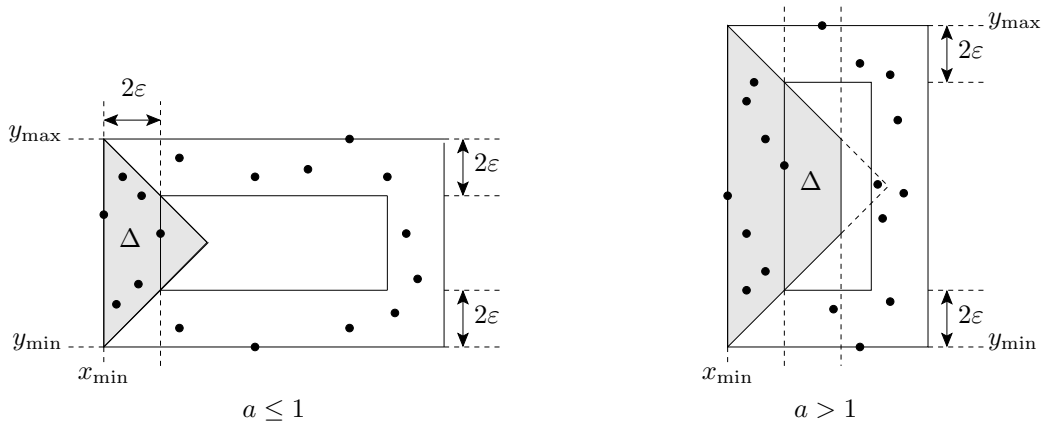

Figure 6: Algorithm 3.

Lemma 8 Algorithm 3 *finds a narrowest covering frame corresponding to* Case 3*, i. e., with* $O_t = y_{\max}$*,* $O_l = x_{\min}$*,* $O_b = y_{\min}$*, and a point of* P *on the inner left edge, if such a frame exists. Its running time is in* $O(n)$*.*

Proof. The running time is obvious since the maximum X-coordinate in Δ can be found in $O(n)$ time and the test if a certain point is covered by the frame takes only constant time.

The correctness is proven as follows. The outer top, left, and bottom edges are fixed, and a point of P must lie on the inner left edge. Clearly, this point can only be contained in the described isosceles, right-angled triangle. For $a > 1$ we can also use the upper bound

$$\varepsilon_{\max} = \frac{O_h}{4} \min\left(1, \frac{2}{1+a}\right) = \frac{O_h}{2(1+a)}$$

for ε from Lemma 5, so there is an upper bound of

$$x_{\min} + 2\varepsilon_{\max} = O_l + \frac{O_h}{1+a}$$

for the X-coordinate of the inner left edge, and Δ must be cut to a trapezoid in this case, see Figure 6. The figure also shows an example for $a > 1$ where a point in the triangle but not in Δ is covered by the right part of the frame.

Furthermore, the only point in Δ that can touch the inner left edge of a covering frame is the one with maximum X-coordinate! This is true because otherwise some points would fall into the inner frame and would not be covered since for any ε between 0 and $\varepsilon_{\max}$ the inner right edge lies to the right of the separating vertical line at $X = O_l + \frac{O_h}{1+a}$, by (3). □

Theorem 9 *A best fitting rectangle with a given aspect ratio, a, for n points in the plane can be computed in time $O(n \log n)$.*

Proof. The following algorithm will compute a best fitting rectangle with aspect ratio a for our point set P.

> Let P_{90}, P_{180}, and P_{270} be the set P rotated by 90°, 180°, resp. 270° about the origin.
> Check all possible cases: From the calls of the following list remember the result with the smallest ε.

	Checks this case:
Call Algorithm 1 for P and a.	
Call Algorithm 1 for P_{90} and $1/a$.	
Call Algorithm 2 for P and a.	
Call Algorithm 2 for P_{90} and $1/a$.	
Call Algorithm 3 for P and a.	
Call Algorithm 3 for P_{90} and $1/a$.	
Call Algorithm 3 for P_{180} and a.	
Call Algorithm 3 for P_{270} and $1/a$.	

Rotate the result with the smallest ε back to the original orientation, this frame is a narrowest covering frame and represents a best fitting rectangle.

For the orientations of 90° and 270°, the aspect ratio is $1/a$ because the roles of height and width are interchanged.

The running time of the algorithm is clearly in $O(n \log n)$, see Lemmas 6 through 8. In these lemmas we have also seen that the Algorithms 1 to 3 find the best fitting rectangles in the respective cases, and by Lemma 4 we know that one of these cases must be a best fitting one. Remark that Algorithm 3 must be called four times to check all four orientations while Algorithm 1 treats two orientations simultaneously, and Case 2 is symmetric thus Algorithm 2 is also called only twice. □

References

[1] P. K. Agarwal, B. Aronov, and M. Sharir. Computing envelopes in four dimensions with applications. *SIAM J. Comput.*, 26:1714–1732, 1997.

[2] P. K. Agarwal and M. Sharir. Efficient randomized algorithms for some geometric optimization problems. *Discrete Comput. Geom.*, 16:317–337, 1996.

[3] P. K. Agarwal, M. Sharir, and S. Toledo. Applications of parametric searching in geometric optimization. *J. Algorithms*, 17:292–318, 1994.

[4] A. Aggarwal and S. Suri. Fast algorithms for computing the largest empty rectangle. In *Proc. 3rd Annu. ACM Sympos. Comput. Geom.*, pages 278–290, 1987.

[5] M. J. Atallah and G. N. Frederickson. A note on finding a maximum empty rectangle. *Discrete Appl. Math.*, 13(1):87–91, 1986.

[6] G. Barequet, A. J. Briggs, M. T. Dickerson, and M. T. Goodrich. Offset-polygon annulus placement problems. *Comput. Geom. Theory Appl.*, 11:125–141, 1998.

[7] G. Barequet, M. T. Dickerson, and M. T. Goodrich. Voronoi diagrams for convex polygon-offset distance functions. *Discrete Comput. Geom.*, 25(2):271–291, 2001.

[8] M. Bern and D. Goldberg. Paper position sensing. In *Proc. 18th Annu. ACM Sympos. Comput. Geom.*, pages 74–81, 2002.

[9] J. Chaudhuri and S. C. Nandy. Largest empty rectangle among a point set. *J. Algorithms*, 46(1):54–78, 2003.

[10] B. Chazelle, R. L. Drysdale, III, and D. T. Lee. Computing the largest empty rectangle. *SIAM J. Comput.*, 15:300–315, 1986.

[11] K. Daniels, V. Milenkovic, and D. Roth. Finding the largest area axis-parallel rectangle in a polygon. *Comput. Geom. Theory Appl.*, 7:125–148, 1997.

[12] M. de Berg, P. Bose, D. Bremner, S. Ramaswami, and G. Wilfong. Computing constrained minimum-width annuli of point sets. *Comput. Aided Design*, 30(4):267–275, Apr. 1998.

[13] J. M. Díaz-Báñez, F. Hurtado, H. Meijer, D. Rappaport, and J. A. Sellarès. The largest empty annulus problem. In *Proc. International Conference Computational Science*, volume 2331 of *Lecture Notes Comp. Science*, pages 46–54, 2002.

[14] J. M. Díaz-Báñez, J. A. Mesa, and A. Schöbel. Continuous location of dimensional structures. Report in Wirtschaftsmathematik 79, Department of Mathematics, Universität Kaiserslautern, Germany, 2002.

[15] Z. Drezner and H. W. Hamacher, editors. *Facility Location: Applications and Theory*. Springer-Verlag, Berlin, Germany, 2002.

[16] C. A. Duncan, M. T. Goodrich, and E. A. Ramos. Efficient approximation and optimization algorithms for computational metrology. In *Proc. 8th ACM-SIAM Sympos. Discrete Algorithms*, pages 121–130, 1997.

[17] H. Ebara, N. Fukuyama, H. Nakano, and Y. Nakanishi. Roundness algorithms using the Voronoi diagrams. In *Abstracts 1st Canad. Conf. Comput. Geom.*, page 41, 1989.

[18] L. W. Foster. *GEO-METRICS II: The application of geometric tolerancing techniques*. Addison-Wesley, 1982.

[19] L. W. Foster. *Geom-Metrics IIIm: The Metric Application of Geometric Dimensioning and Tolerancing Techniques*. Prentice Hall, 1993.

[20] J. García-López, P. Ramos, and J. Snoeyink. Fitting a set of points by a circle. *Discrete Comput. Geom.*, 20:389–402, 1998.

[21] L. J. Guibas and R. Seidel. Computing convolutions by reciprocal search. *Discrete Comput. Geom.*, 2:175–193, 1987.

[22] M. E. Houle and G. T. Toussaint. Computing the width of a set. *IEEE Trans. Pattern Anal. Mach. Intell.*, PAMI-10(5):761–765, 1988.

[23] N. Megiddo. Applying parallel computation algorithms in the design of serial algorithms. *J. ACM*, 30(4):852–865, 1983.

[24] A. Mukhopadhyay and S. V. Rao. On computing a largest empty arbitrarily oriented rectangle. *Internat. J. Comput. Geom. Appl.*, 13(3):257–271, 2003.

[25] A. Naamad, W.-L. Hsu, and D. T. Lee. On the maximum empty rectangle problem. *Discrete Appl. Math.*, 8:267–277, 1984.

[26] M. Orlowski. A new algorithm for the largest empty rectangle problem. *Algorithmica*, 5:65–73, 1990.

[27] T. J. Rivlin. Approximating by circles. *Computing*, 21:93–104, 1979.

[28] U. Roy and X. Zhang. Establishment of a pair of concentric circles with the minimum radial separation for assessing roundness error. *Comput. Aided Design*, 24(3):161–168, 1992.

[29] M. Smid and R. Janardan. On the width and roundness of a set of points in the plane. *Internat. J. Comput. Geom. Appl.*, 9:97–108, 1999.

[30] A. D. Wainstein. A non-monotonous placement problem in the plane. In *Software Systems for Solving Optimal Planning Problems, 9th All-Union Symp. Minsk, BSSR, USSR, Symp. Abstracts*, pages 70–71, 1986.

[31] R. K. Walker and V. Srinivasan. Creation and evolution of the ASME Y14.5.1M standard. *Manufacturing Review*, 7(4):16–23, 1994.