

Largest Inscribed Rectangles in Convex Polygons*

Christian Knauer[†] Lena Schlipf[‡] Jens M. Schmidt[§]
Hans Raj Tiwary[¶]

Abstract

We consider approximation algorithms for the problem of computing an inscribed rectangle having largest area in a convex polygon on n vertices. If the order of the vertices of the polygon is given, we present a randomized algorithm that computes an inscribed rectangle with area at least $(1 - \epsilon)$ times the optimum with probability t in time $O(\frac{1}{\epsilon} \log n)$ for any constant $t < 1$. We further give a deterministic approximation algorithm that computes an inscribed rectangle of area at least $(1 - \epsilon)$ times the optimum in running time $O(\frac{1}{\epsilon^2} \log n)$ and show how this running time can be slightly improved.

1 Introduction

Much work has been devoted in the past to compute inscribed objects of maximum area in polygons. Most contributions focus on objects that are again polygons, e. g., largest axis-aligned rectangles in convex or non-convex polygons [2, 5], largest squares and equilateral triangles in convex polygons [6], and largest empty rectangles on point sets [4].

Given a convex polygon P with n vertices, we want to compute a largest inscribed rectangle in P . To our knowledge, no optimal algorithm for this problem is published so far, although there is a straight-forward way in time $O(n^4)$ (we will give a sketch in the next chapter). Moreover, we are not aware of an algorithm that computes a $(1 - \epsilon)$ -approximation for this problem. Ahn et al. describe how to approximate axially symmetric polygons in the more general class of convex sets [1], however the computed polygon is not necessarily a rectangle. Hall-Holt et al. restrict the problem

*Work by Schlipf and Schmidt was partly supported by the Deutsche Forschungsgemeinschaft within the research training group “Methods for Discrete Structures” (GRK 1408).

[†]Institute of Computer Science, Universität Bayreuth, Germany, christian.knauer@uni-bayreuth.de.

[‡]Institute of Computer Science, Freie Universität Berlin, Germany, schlipf@inf.fu-berlin.de.

[§]MPI für Informatik, Saarbrücken, Germany, jens.schmidt@mpi-inf.mpg.de.

[¶]Département de Mathématique, Université Libre de Bruxelles, Belgium, hans.raj.tiwary@ulb.ac.be.

to *fat* rectangles [7], i. e., rectangles with an aspect ratio that is bounded by a constant. Under the assumption that a largest inscribed rectangle is fat, they $(1 - \epsilon)$ -approximate the largest fat rectangle in simple polygons in time $O(n)$; in polygons with holes, their approximation algorithm runs in $O(n \log n)$ time.

We obtain approximation algorithms for general largest rectangles with running times that are only logarithmically dependent on n , if the ordering of the vertices of the convex polygon P is given. The assumption on the vertex ordering is common when handling polygons: In fact, Alt et al. show under this assumption that the largest axis-aligned inscribed rectangle inside a convex polygon can be computed in logarithmic time [2]. If not already given, the ordering can be computed using standard convex hull algorithms in $O(n \log n)$ time. We will assume throughout the paper that the ordering is given. Our results show that fatness is not required for approximating a largest inscribed rectangle.

The main result of this paper can be stated as follows.

Theorem 1. *Let P be a convex polygon with n vertices. Suppose the vertices of the polygon are given in clockwise order. Then, an inscribed rectangle in P with area of at least $(1 - \epsilon)$ times the area of a largest inscribed rectangle can be computed*

- *with probability t in $O(\frac{1}{\epsilon} \log n)$ deterministic time for any constant $t < 1$.*
- *in $O(\frac{1}{\epsilon^2} \log n)$ deterministic time.*
- *in $O(\frac{1}{\epsilon} \log \frac{1}{\epsilon} \log n + \frac{1}{\epsilon^{28}})$ deterministic time.*

2 Preliminaries

We denote the area of a polygon P by $|P|$. A line segment connecting two points a and b is denoted by \overline{ab} and its length by $|\overline{ab}|$. For a given convex polygon P , let R_{opt} be a largest inscribed rectangle. Note that in general the largest inscribed rectangle is not unique; we will use R_{opt} to denote any one of the largest inscribed rectangles.

A slow but optimal algorithm. We briefly sketch an optimal algorithm with running time $O(n^4)$. If a corner of R_{opt} coincides with a vertex of P , we call it a *vertex-corner*; if it is contained in a boundary edge of P , we call it an *edge-corner* (note that an edge-corner can be a vertex-corner). It is easy to prove that R_{opt} is either a square with two opposite vertex-corners or contains at least 3 edge-corners. The largest square in a convex polygon can be computed in $O(n^2)$ time [6]. For the latter case, let e_1, e_2, e_3 be boundary

edges of P containing the edge-corners. If the fourth corner is also an edge-corner, let e_4 be an edge containing it; otherwise, let e_4 be a boundary edge that is intersected by extending the longer side of R_{opt} . For each of the $\binom{n}{4}$ possible edge sets $\{e_1, \dots, e_4\}$, we compute all largest area rectangles with corners in at least 3 of the edges e_1, \dots, e_4 such that the possibly leftover edge is intersected by the extension of the largest rectangle side. This is a problem of constant size, as it can be expressed as optimization problem with a constant number of variables and polynomial inequalities of constant degree each. Solving this problem takes therefore constant time, which gives the total running time $O(n^4)$.

We want to approximate the largest inscribed rectangle in a convex polygon. This will allow us to obtain exponentially better running times for every fixed approximation ratio. If we know the direction d_{opt} of one of the sides of R_{opt} , we can compute the largest rectangle R_{opt} itself in $O(\log n)$ time by applying the algorithm of Alt et al. [2]. The general idea of our algorithm is to approximate the direction of alignment of a largest inscribed rectangle and to prove that the area of the largest inscribed rectangle aligned along this direction also approximates $|R_{opt}|$. For the computation, we construct a set of candidate directions and find the largest inscribed rectangle along each of these directions using the algorithm of Alt et al. [2]. The number of candidate directions will be $O(\frac{1}{\epsilon})$ for the randomized version of our algorithm, and $O(\frac{1}{\epsilon^2})$ or $O(\frac{1}{\epsilon} \log \frac{1}{\epsilon})$ for the deterministic one.

3 Approximating the direction of R_{opt}

We want to find a direction close enough to the direction of any side of R_{opt} . This direction will be called an ϵ -close direction, for a fixed $\epsilon > 0$. To define what ϵ -close means, first suppose that we know R_{opt} and denote the intersection of its diagonals as its *center* s . Let \overline{ab} be one of the two shortest sides of R_{opt} and let d be the midpoint of the segment \overline{ab} , see Figure 1. Then $\angle(asb) \leq \frac{\pi}{2}$ and we can define the triangles T_1 and T_2 as the two triangles with vertices s , d and the third vertex being either $f_1 := d + \epsilon(b - d)$ or $f_2 := d - \epsilon(b - d)$. Analogously, choosing the side of R_{opt} opposite of \overline{ab} gives the two triangles T_3 and T_4 having the same area. The area for each triangle T_i is $\epsilon|\overline{db}||\overline{sd}|/2$ and therefore an $\epsilon/8$ -fraction of $|R_{opt}| = 4|\overline{db}||\overline{sd}|$. We define a direction to be ϵ -close if the line containing s with that direction intersects $\overline{f_1f_2}$.

Lemma 2. *A largest inscribed rectangle R_{app} that is aligned to an ϵ -close direction contains an area of at least $(1 - 8\epsilon)|R_{opt}|$.*

Consider the triangle $T = asb$ in R_{opt} (see Figure 2) and an ϵ -close direction d_{app} . Let ℓ be the line with direction d_{app} that contains s ; we

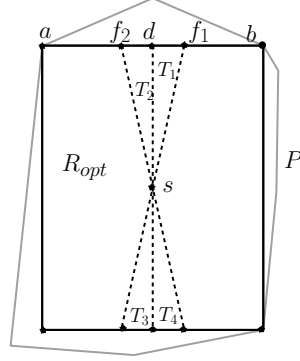


Figure 1: A largest rectangle R_{opt} in a convex polygon P . The area for each T_i , $1 \leq i \leq 4$, is an $\epsilon/8$ -fraction of $|R_{opt}|$.

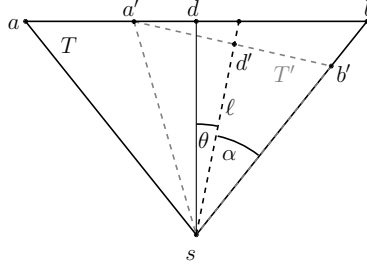


Figure 2: The triangle $T = asb$.

assume w.l.o.g. that ℓ intersects $\overline{df_1}$ in Figure 1, as the case for $\overline{df_2}$ is symmetric. We denote the angle between \overline{sd} and d_{apx} by θ and the angle between d_{apx} and \overline{sb} by α . Let T' be the isosceles triangle of maximum area that is contained in T and symmetric along ℓ (i.e., with ℓ as perpendicular bisector of the base side). Note that T' must contain s as vertex. Let a' and b' be the two remaining vertices of T' and consider the midpoint d' of the segment $\overline{a'b'}$.

Instead of comparing $|R_{apx}|$ with $|R_{opt}|$ directly, we now compare the triangles T and T' . If we can show that $|T'| \geq (1 - c\epsilon)|T|$ for some constant c , then the largest rectangle aligned to d_{apx} has at least an area of $(1 - c\epsilon)|R_{opt}|$. The reduction to triangles does not matter for the approximation, as $|R_{opt}| = 4|T|$ and $|R_{apx}| \geq 4|T'|$.

Recalling elementary trigonometry we see that

$$\begin{aligned} \epsilon &\geq \frac{\tan \theta}{\tan(\theta + \alpha)}, & (*) \\ |T| &= |\overline{sa}|^2 \sin(\alpha + \theta) \cos(\alpha + \theta), \text{ and} \\ |T'| &= |\overline{sa'}|^2 \sin(\alpha) \cos(\alpha) = |\overline{sa}|^2 \frac{\cos^2(\alpha + \theta)}{\cos^2(\alpha - \theta)} \sin(\alpha) \cos(\alpha). \end{aligned}$$

On the other hand, we want to show that

$$\begin{aligned}
& |T'| \geq (1 - c\epsilon)|T| \\
& \Leftrightarrow \frac{|T'|}{|T|} \geq 1 - c\epsilon \\
& \Leftrightarrow \frac{|\bar{s}a|^2 \frac{\cos^2(\alpha+\theta)}{\cos^2(\alpha-\theta)} \sin(\alpha) \cos(\alpha)}{|\bar{s}a|^2 \sin(\alpha + \theta) \cos(\alpha + \theta)} \geq 1 - c\epsilon \\
& \Leftrightarrow \frac{\sin(\alpha) \cos(\alpha)}{\cos^2(\alpha - \theta) \tan(\alpha + \theta)} \geq 1 - c\epsilon
\end{aligned}$$

for a constant c . We use the following lemma.

Lemma 3. *The function $\frac{\sin(\alpha)\cos(\alpha)}{\cos^2(\alpha-\theta)} + c \tan(\theta) - \tan(\theta + \alpha)$ is positive for $0 \leq \alpha \leq \frac{\pi}{4}, 0 \leq \theta \leq \frac{\pi}{8}$ and any constant $c \geq 8$.*

To prove this lemma, we need the following two propositions.

Proposition 4. $\frac{1}{4} \tan(x) \leq \tan(\frac{x}{3})$ for $0 \leq x \leq \frac{\pi}{4}$.

Proof. Consider the function $f(x) = \frac{1}{4} \tan(x) - \tan(\frac{x}{3})$. We have to show that $f(x) \leq 0$ for $0 \leq x \leq \frac{\pi}{4}$. The first and second derivatives of $f(x)$ with respect to x are:

$$\begin{aligned}
f'(x) &= \frac{1}{4} \sec^2(x) - \frac{1}{3} \sec^2\left(\frac{x}{3}\right), \\
f''(x) &= \frac{1}{2} \sec^2(x) \tan(x) - \frac{2}{9} \sec^2\left(\frac{x}{3}\right) \tan\left(\frac{x}{3}\right)
\end{aligned}$$

Since $\tan(x) \geq \tan(\frac{x}{3})$, we have

$$f''(x) \geq 2 \tan\left(\frac{x}{3}\right) \left(\frac{1}{4} \sec^2(x) - \frac{1}{9} \sec^2\left(\frac{x}{3}\right)\right).$$

Let x' be the root of $f'(x) = 0$. That is, let $x' \in [0, \frac{\pi}{4}]$ be such that

$$\frac{1}{4} \sec^2(x') - \frac{1}{3} \sec^2\left(\frac{x'}{3}\right) = 0$$

Since $\tan(x) \geq 0$ in the domain $0 \leq x \leq \frac{\pi}{4}$, we have

$$f''(x') \geq 2 \tan\left(\frac{x'}{3}\right) \frac{2}{9} \sec^2\left(\frac{x'}{3}\right) \geq 0$$

Therefore in the range $[0, \frac{\pi}{4}]$, $f(x)$ attains minima whenever $f'(x) = 0$ and the maxima are attained only at the boundary. Since $f(0) = 0$ and $f(\frac{\pi}{4}) < 0$, $f(x) \leq 0$ for $0 \leq x \leq \frac{\pi}{4}$. \square

Proposition 5. *If we choose $\epsilon \leq \frac{1}{4}$, then $\theta \leq \frac{\alpha}{2}$.*

Proof. This proof uses Proposition 4 for the inequality marked with (\times).

$$\begin{aligned} \frac{\tan(\theta)}{\tan(\alpha + \theta)} &\leq \epsilon \leq \frac{1}{4} \\ \tan(\theta) &\leq \frac{1}{4} \tan(\alpha + \theta) \stackrel{(\times)}{\leq} \tan\left(\frac{\theta + \alpha}{3}\right) \\ \theta &\leq \frac{\theta + \alpha}{3} \text{ for } [0, \frac{\pi}{4}] \\ \theta &\leq \frac{\alpha}{2} \end{aligned}$$

□

Now we can continue with the proof of Lemma 3.

Proof of Lemma 3.

$$\begin{aligned} &\frac{\sin(\alpha) \cos(\alpha)}{\cos^2(\alpha - \theta)} + c \tan(\theta) - \tan(\theta + \alpha) \\ = &\tan(\alpha) \frac{\cos^2(\alpha)}{\cos^2(\alpha - \theta)} + c \tan(\theta) - \frac{\tan(\theta)}{1 - \tan(\theta) \tan(\alpha)} - \frac{\tan(\alpha)}{1 - \tan(\theta) \tan(\alpha)} \end{aligned}$$

Since $\frac{1}{1 - \tan(\theta) \tan(\alpha)} \leq \frac{1}{1 - \tan(\frac{\pi}{8})}$, it suffices to show that

$$\tan(\alpha) \frac{\cos^2(\alpha)}{\cos^2(\alpha - \theta)} + c \tan(\theta) - \frac{\tan(\alpha)}{1 - \tan(\theta) \tan(\alpha)} \geq 0$$

for some constant $c' = c - 1.71 > 0$.

$$\begin{aligned} &\tan(\alpha) \frac{\cos^2(\alpha)}{\cos^2(\alpha - \theta)} + c' \tan(\theta) - \frac{\tan(\alpha)}{1 - \tan(\theta) \tan(\alpha)} \\ = &\tan(\alpha) \left(\frac{1 + \frac{(\tan(\alpha) - \tan(\theta))^2}{(1 + \tan(\theta) \tan(\alpha))^2}}{1 + \tan^2(\alpha)} + c' \frac{\tan(\theta)}{\tan(\alpha)} - \frac{1}{1 - \tan(\theta) \tan(\alpha)} \right) \end{aligned}$$

Replacing $\tan(\theta)$ by x and $\tan(\alpha)$ by y , we want to show that

$$\begin{aligned} &y \left(\frac{1 + \frac{(y-x)^2}{(1+xy)^2}}{1+y^2} + c' \frac{x}{y} - \frac{1}{1-xy} \right) > 0 \\ &y \left(\frac{1 + \frac{(y-x)^2}{(1+xy)^2}}{1+y^2} + c' \frac{x}{y} - \frac{1}{1-xy} \right) \\ = &\frac{y(1-xy)(1+x^2) + c'x(1+xy)^2(1-xy) - y(1+xy)^2}{(1+xy)^2(1-xy)} \end{aligned}$$

Then, it suffices to show that

$$y(1 - xy)(1 + x^2) + c'x(1 + xy)^2(1 - xy) - y(1 + xy)^2 > 0$$

Expanding the left-hand side gives the equivalent inequality

$$x((c' + 1)xy - 3y^2 - x^2y^2 + c' + c'x^2y^2 - c'x^3y^3 - 2c'x^2y^2 - xy^3) > 0$$

Since $\tan(\alpha) \leq 1$, it suffices to show that $(-3 - x^2 + c' - c'x^3 - 2c'x^2 - x) > 0$. That is $(c' - 3 - x - (2c' + 1)x^2 - c'x^3) > 0$. Since $x = \tan(\theta) \leq \tan(\frac{\pi}{8})$ it suffices to pick c' such that $c' - 3 - \tan(\frac{\pi}{8}) - (2c' + 1)\tan^2(\frac{\pi}{8}) - c'\tan^3(\frac{\pi}{8}) > 0$. Thus $c' > \frac{3 + \tan(\frac{\pi}{8}) + \tan^2(\frac{\pi}{8})}{1 - 2\tan^2(\frac{\pi}{8}) - \tan^3(\frac{\pi}{8})} \approx 6.12$. This implies that the function $\frac{\sin(\alpha)\cos(\alpha)}{\cos^2(\alpha - \theta)} + c \tan(\theta) - \tan(\theta + \alpha)$ is positive for $0 \leq \alpha \leq \frac{\pi}{4}, 0 \leq \theta \leq \frac{\pi}{8}$ and for any $c \geq 8$.

Using Lemma 3 with (*), we obtain the following corollary.

Corollary 6. Let $0 \leq \alpha \leq \frac{\pi}{4}, 0 \leq \theta \leq \frac{\pi}{8}$ and let $c \geq 8$. Then $\frac{|T'|}{|T|} = \frac{\sin(\alpha)\cos(\alpha)}{\cos^2(\alpha - \theta)\tan(\alpha + \theta)} \geq 1 - c \frac{\tan \theta}{\tan(\theta + \alpha)} \geq 1 - c\epsilon$.

Corollary 6 shows that $|T'| \geq (1 - 8\epsilon)|T|$, implying that $|R_{apx}| \geq (1 - 8\epsilon)|R_{opt}|$.

4 How to get an ϵ -close direction

It only remains to show how to compute a direction ϵ -close to d_{opt} efficiently. Assume first that we know the center s of R_{opt} . If we choose $\Theta(\frac{1}{\epsilon})$ random points uniformly distributed inside P , at least one of them lies with high probability in one of the triangles T_1, T_2, T_3 and T_4 . Thus, taking the direction from s to this point gives us immediately an ϵ -close direction (see Figure 1). As we do not have the information about the location of s , assume that any other point p inside R_{opt} is given. Then at least one triangle $T_i, 1 \leq i \leq 4$, has a translated copy T'_i , where the translation maps s to p . The triangle T'_i is contained in R_{opt} and therefore also contained in P . Picking a point q' in T'_i and taking the direction $\overline{pq'}$ has the same effect as picking a point q in T_i and taking the direction \overline{sq} . Thus, we do not have to compute s explicitly. Instead, it is sufficient to find a point inside R_{opt} .

Even though we do not know R_{opt} , picking points from it essentially amounts to picking points from the input polygon because the area of the largest inscribed rectangle in a convex polygon is at least a constant factor of the area of the polygon. More formally,

Lemma 7 ([10]). Let P be a convex polygon and R_{opt} be a largest inscribed rectangle in P , then $|R_{opt}| \geq |P|/2$.

4.1 Randomized algorithm

It follows from Lemma 7 that if we pick k points sampled uniformly at random from a convex polygon P , the expected number of points inside R_{opt} is $\frac{k}{2}$. All these points are distributed uniformly at random inside R_{opt} . Moreover, if we pick $\Theta(\frac{1}{\epsilon})$ points uniformly at random, the expected number of points inside the triangle T'_i is $\Theta(1)$. This obtains a constant success probability, which can be increased to an arbitrary high constant $t < 1$ by probability amplification, without decreasing the asymptotic running time. Thus, we have the following lemma.

Lemma 8. *Let a convex polygon P and a source of random points in P be given. Then we can compute a $(1 - \epsilon)$ -approximation R_{apx} for the largest inscribed rectangle in P with probability t in time $O(\frac{1}{\epsilon} \log n)$ for an arbitrary constant $t < 1$.*

We can achieve the same running time without random points in P being given. It is easy to see that with a preprocessing of $O(n \log n)$ we can create a data structure for a (not necessarily convex) polygon P that returns a point distributed uniformly at random inside P in $O(\log n)$ time per sample. This can be achieved by first computing a triangulation of the point set and then creating a balanced binary tree with the triangles as leaves, where the weight of any node is the sum of areas of all triangles contained in the subtree rooted at that node. Sampling a random point from P then amounts to traversing this tree from root to a leaf and following the left or the right child at any node with the probability proportional to their weights.

Since the ordering of the vertices of P is given and we want to avoid any preprocessing for P , we will not sample points from P uniformly at random. Instead, we take a uniform distribution over a square and fit these points inside the polygon. Thus, the sampling from P will simulate the sampling of random points from a square. Let v_t, v_b be the topmost and the bottommost vertices of P ; their computation takes $O(\log n)$ time. We pick a height h between the two vertices uniformly at random and take the longest horizontal segment that fits inside P at this height. Again, this segment can be computed in $O(\log n)$ time. We pick a point uniformly at random on this segment. This will be our sample point in P . We can repeat this process as many times as desired to get a large set of sample points that are in P . Each of these sample points can be generated in $O(\log n)$ time assuming that the ordering of vertices of P is known in advance.

We show that such a sampling works for our algorithm. Recall that two points p and q from P are needed such that p lies in a largest inscribed rectangle R_{opt} and q lies in a triangle of area $\Omega(\epsilon)$ that is a translated copy of one of the T_i 's (see Figure 1). With our sampling method, the probability that a sample point is contained in any convex region Q of area $\epsilon|P|$ will be at least $\frac{\epsilon}{2}$.

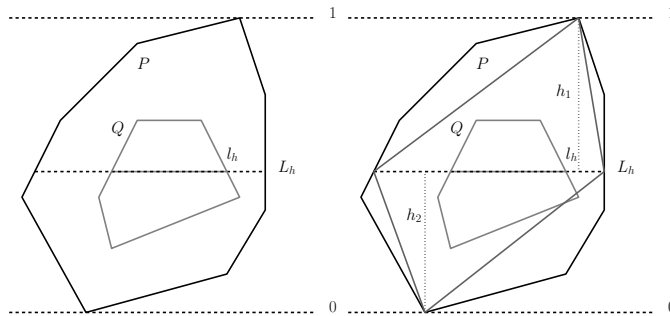


Figure 3: Sampling from a convex region Q in P .

Let L_h be the length of the largest horizontal segment inside P at height h , and l_h be the length of the largest horizontal segment inside Q at height h . Also, assume that the bottommost and topmost points in P are at heights 0 and 1 respectively (see Figure 3). Then $\frac{|Q|}{|P|} = \frac{\int_0^1 l_h dh}{\int_0^1 L_h dh}$. The probability that a sample point using the above sampling method lies in Q is $\int_0^1 \frac{l_h}{L_h} dh$. For any value of h , we can find a quadrilateral that fits inside P and has area at least $\frac{L_h}{2}$. This implies that $\frac{L_h}{2} \leq \int_0^1 L_h dh$ and

$$\int_0^1 \frac{l_h}{L_h} dh \geq \frac{1}{2} \frac{\int_0^1 l_h dh}{\int_0^1 L_h dh}.$$

Since each of these sample points can be generated in logarithmic time, the complexity of our algorithm is $O(\frac{1}{\epsilon} \log n)$. We summarize the steps in Algorithm 1.

Algorithm 1

- 1: Take $\Theta(1)$ points in P with the aforementioned distribution and store them in U .
 - 2: Take $\Theta(1/\epsilon)$ points in P with the aforementioned distribution and store them in V .
 - 3: $|R_{\text{apx}}| = 0$
 - 4: **for all** $u \in U$ **do**
 - 5: **for all** $v \in V$ **do**
 - 6: Compute the largest inscribed rectangle S that is aligned to \overline{uv} .
 - 7: **if** $|S| \geq |R_{\text{apx}}|$ **then**
 - 8: $R_{\text{apx}} = S$
 - return** R_{apx}
-

4.2 Deterministic algorithm

We begin by summarizing the results of this section.

Lemma 9. *Let a convex polygon P and the cyclic order of vertices in P be given. Then we can compute a $(1 - \epsilon)$ -approximation R_{apx} for the largest inscribed rectangle in P in $O(\frac{1}{\epsilon} \log \frac{1}{\epsilon} \log n + \frac{1}{\epsilon^{28}})$ time.*

For the deterministic case, it remains to show how the algorithm computes sample points in P . First, we compute an enclosing rectangle R_ϵ of P such that $|R_\epsilon|$ is only a constant factor times bigger than $|P|$. This can be done using the following lemma due to Ahn et al. [1].

Lemma 10 ([1, Lemma 5]). *Let P be a convex polygon with n vertices given in a cyclic order. Then there is an algorithm that computes an enclosing rectangle R such that $P \subset R$ and $|R| \leq 2\sqrt{2}|P|$ in $O(\log n)$ time.*

Creating a grid of constant size in an enclosing rectangle R of Lemma 10 allows us to ensure a constant number of grid points in P . This is proven in Lemma 12 by using Pick's theorem.

Theorem 11 (Pick's Theorem [9]). *Let an integer grid and a simple polygon P with all vertices lying on the grid points be given. Let i be the number of grid points contained in P and b be the number of grid points on the boundary of P . Then $|P| = \frac{b}{2} + i - 1$.*

Lemma 12. *For a convex set S , a constant c , and every enclosing rectangle R of S with $|R| \leq c|S|$, S contains at least $\frac{k^2}{2c}$ grid points of a $k \times k$ grid on R for k being a sufficiently large constant ($k \geq 8c$).*

Proof. Let G be a $k \times k$ grid on R and let $|R| = 1$. We shrink S to a maximum area polygon $S' \subseteq S$ having all vertices on grid points of G . Because of convexity, $|S| - |S'|$ is at most the area of $4k$ grid cells.

$$\begin{aligned} |S'| &\geq |S| - 4k \frac{1}{k^2} \geq \frac{1}{c} - \frac{4}{k} \\ |S'| &= \left(\frac{b}{2} + i - 1\right) \frac{1}{k^2} \geq \frac{1}{c} - \frac{4}{k} \quad (\text{by Pick's theorem}) \\ b + i &\geq \frac{k^2}{c} - 4k + 1 \end{aligned}$$

Thus, at least $\frac{1}{2c}k^2$ grid points lie in S , for k being a sufficiently large constant ($k \geq 8c$). \square

It follows from Lemma 12 that choosing a grid with constant size on the rectangle R implies that R_{opt} in P contains a constant number of grid points (in fact $\frac{k^2}{2c}$). For the next algorithm (Algorithm 2), we will only use one of these grid points. Additionally, Lemma 12 shows that every ϵ -fraction of P , in particular every triangle T'_i , contains many grid points for a big enough grid on R . This fact will be used later to improve the running time of the algorithm with ϵ -nets.

Algorithm 2

```
1: Compute an enclosing rectangle  $R_e$  with area  $|R_e| \leq 2\sqrt{2}|P|$ 
2: Compute a  $\Theta(1) \times \Theta(1)$  grid on  $R_e$ . Let  $G_1$  be the set of grid points.
3: Compute a  $\Theta(\frac{1}{\epsilon}) \times \Theta(\frac{1}{\epsilon})$  grid on  $R_e$ . Let  $G_2$  be the set of grid points.
4:  $|R_{\text{apx}}| = 0$ 
5: for all  $u \in G_1$  do
6:   for all  $v \in G_2$  do
7:     Compute the largest inscribed rectangle  $S$  that is aligned to  $\overline{uv}$ 
8:     if  $|S| \geq |R_{\text{apx}}|$  then
9:        $R_{\text{apx}} = S$ 
return  $R_{\text{apx}}$ 
```

The idea is to take two grids G_1 and G_2 on R of size $\Theta(1) \times \Theta(1)$ and $\Theta(\frac{1}{\epsilon}) \times \Theta(\frac{1}{\epsilon})$, respectively, iterate through all pairs of grid points in $G_1 \times G_2$ and get at least one pair (u, v) with $u \in R_{\text{opt}}$ and $v \in T'_i$ using Lemma 12. Hence, the direction of \overline{uv} is ϵ -close. We summarize the steps in Algorithm 2.

Algorithm 2 is deterministic with running time $O(\frac{1}{\epsilon^2} \log n)$. We can further reduce the running time to $O(\frac{1}{\epsilon} \log \frac{1}{\epsilon} \log n + \frac{1}{\epsilon^{28}})$ by using the tools from the theory of ϵ -nets. Here we just give an outline of how these tools can be used, and we refer the reader to [8] for more details.

A subset S' of a given set S of N points is called an ϵ -net for S with respect to a set of objects, if any object containing at least $\frac{\epsilon}{2}N$ points of S contains a point of S' . For objects with VC-dimension d , a subset S' of size $O(\frac{1}{\epsilon} \log \frac{1}{\epsilon})$ always exists and can be computed in deterministic time $O(N^{2d})$. Triangles have VC-dimension 7, and we consider the set S of grid points of a $\frac{1}{\epsilon} \times \frac{1}{\epsilon}$ grid, so $N = \frac{1}{\epsilon^2}$. Thus, we can compute an ϵ -net for S of size $O(\frac{1}{\epsilon} \log \frac{1}{\epsilon})$ in time $O(\frac{1}{\epsilon^{28}})$.

5 Largest inscribed rectangles in simple polygons

The same ideas can be used to approximate the largest inscribed rectangle in a simple polygon with or without holes. It is easy to see that the largest inscribed rectangle R_{opt} in a simple polygon (with or without holes) on n vertices has an area of at least $\frac{1}{2(n-2)}$ times the area of the polygon. Moreover, a largest axis-aligned rectangle in a simple polygon can be computed in $O(n \log n)$ time [3] and in a simple polygon with holes in $O(n \log^2 n)$ time [5]. Since $|R_{\text{opt}}|$ is an $\Omega(\frac{1}{n})$ -fraction of P and the area of each of the four triangles inside R_{opt} is an $\Omega(\frac{\epsilon}{n})$ -fraction of P , we get the following running times for computing an inscribed rectangle of area at least $(1 - \epsilon)R_{\text{opt}}$.

- For simple polygons: With constant probability in time $O(\frac{1}{\epsilon} n^3 \log n)$.
- For polygons with holes: With constant probability in time $O(\frac{1}{\epsilon} n^3 \log^2 n)$.

In comparison with the algorithm of Hall-Holt et al. [7], which deals only with fat rectangles, our algorithm can handle general rectangles at the expense of a slower running time.

6 Open Problems

One open related problem is to approximate a largest perimeter inscribed rectangle. Our algorithms base on the fact that the area of a largest area inscribed rectangle has constant fraction of the area of the polygon itself. This is not the case for a largest perimeter rectangle. Consider a triangle with two angles being strictly smaller than $\pi/4$. Then the largest perimeter rectangle is exactly the largest side of this triangle, hence its area is zero.

Another remaining open problem is to find an efficient exact algorithm for computing a largest area inscribed rectangle in a convex polygon.

Acknowledgments

We thank the anonymous referees for their helpful comments.

References

- [1] H.-K. Ahn, P. Brass, O. Cheong, H.-S. Na, C.-S. Shin, and A. Vigneron. Inscribing an axially symmetric polygon and other approximation algorithms for planar convex sets. *Computational Geometry*, 33(3):152 – 164, 2006.
- [2] H. Alt, D. Hsu, and J. Snoeyink. Computing the largest inscribed isothetic rectangle. In *Proc. 7th Canad. Conf. Comput. Geom.*, pages 67–72, 1995.
- [3] R. P. Boland and J. Urrutia. Finding the largest axis-aligned rectangle in a polygon in $O(n \log n)$ time. In *In Proc. 13th Canad. Conf. Comput. Geom.*, pages 41–44, 2001.
- [4] J. Chaudhuri, S. C. Nandy, and S. Das. Largest empty rectangle among a point set. *J. Algorithms*, 46(1):54–78, 2003.
- [5] 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.
- [6] A. DePano, Y. Ke, and J. O’Rourke. Finding largest inscribed equilateral triangles and squares. In *Proc. 25th Allerton Conf. Commun. Control Comput.*, pages 869–878, Oct. 1987.

- [7] O. Hall-Holt, M. J. Katz, P. Kumar, J. S. B. Mitchell, and A. Sityon. Finding large sticks and potatoes in polygons. In *SODA '06*, 2006.
- [8] J. Matoušek. Approximations and optimal geometric divide-and-conquer. In *STOC'91: Proceedings of the twenty-third annual ACM symposium on Theory of computing*, pages 505–511, 1991.
- [9] G. Pick. Geometrisches zur Zahlenlehre. *Sitzungber. Lotos, Naturwissen Zeitschrift, Prague*, 19:311–319, 1899.
- [10] K. Radziszewski. Sur une problème extrémal relatif aux figures inscrites et circonscrites aux figures convexes. *Ann. Univ. Mariae Curie-Sklodowska, Sect. A6*, pages 5–18, 1952.