Maximum-Area Triangle in a Convex Polygon, Revisited

Article *in* Information Processing Letters · May 2017

DOI: 10.1016/j.ipl.2020.105943

4 authors, including:

Vahideh Keikha
Amirkabir University of Technology
11 PUBLICATIONS **10** CITATIONS

SEE PROFILE

Jérôme Urhausen
Utrecht University
5 PUBLICATIONS **6** CITATIONS

SEE PROFILE

Maximum-Area Triangle in a Convex Polygon, Revisited

Vahideh Keikha[a], Maarten Löffler[b,], Jérôme Urhausen[c], Ivor van der Hoog[b]

[a]*Laboratory of Algorithms and Computational Geometry, Department of Mathematics and Computer Science, Amirkabir University of Technology, Tehran, Iran*
[b]*Department of Information and Computing Sciences, Utrecht University, Utrecht, The Netherlands*
[c]*Department of Informatics, Karlsruhe Institute of Technology, Karlsruhe, Germany*

Abstract

In this note, we revisit the following problem: Given a convex polygon P, find the largest-area inscribed triangle. We show by example that the linear-time algorithm presented by Dobkin and Snyder [1] for solving this problem fails. We then proceed to show that with a small adaptation, their approach does lead to a quadratic-time algorithm. We also present a more involved $O(n \log n)$ time divide-and-conquer algorithm. Finally, we discuss the implications of this discovery on the literature.

Keywords:
Computational Geometry, k-gon, Largest Area Triangle

1. Introduction

We revisit a classic problem in computational geometry: Given a convex polygon P, find the largest-area inscribed triangle. Figure 1 illustrates the problem. Dobkin and Snyder [1] present a linear-time algorithm to solve this problem. In this note, we present an example of a polygon on which their algorithm fails—refer to Figure 3—and carefully analyse the underlying geometry. We give insight into the reason for the failure, and use this to create a $O(n \log n)$ algorithm to solve the problem.

The study of geometric containment problems was initiated by Michael Shamos [2], who considered the question of finding the longest line segment in a convex polygon, otherwise known as the *diameter* of the polygon. He presented a linear-time algorithm in his thesis [3], based on a technique which is now known under the name *rotating calipers* [4] Dobkin and Snyder [1] also present a linear-time algorithm for computing the diameter, which was found to be incorrect by Avis *et al.* [5].

In fact, Dobkin and Snyder [1] claim to have a generic linear-time algorithm for finding the largest inscribed k-gon inside a convex polygon, a claim which was later retracted: Boyce *et al.* [6] observe that the algorithm by Dobkin and Snyder fails for $k = 5$ and instead present

*Maarten Löffler is the corresponding author
Email addresses: va.keikha@aut.ac.ir (Vahideh Keikha), m.loffler@uu.nl (Maarten Löffler), (jerome.urhausen@gmail.com (Jérôme Urhausen), i.d.vanderhoog@uu.nl (Ivor van der Hoog)

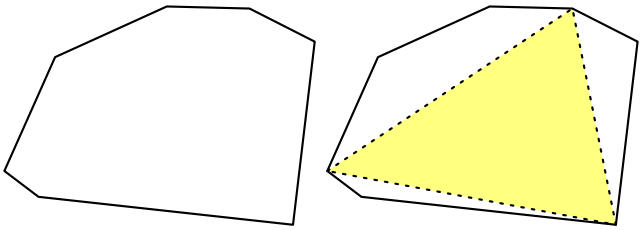


Figure 1: (left) A convex polygon. (right) The largest (by area) inscribed triangle.

a $O(kn \log n + n \log^2 n)$ algorithm. Aggarwal *et al.* [7] improve their result to $O(kn + n \log n)$ time by using a matrix search method. However, both algorithms still rely on the correctness of the Dobkin and Snyder algorithm for triangles, and hence, also fail by our analysis.

1.1. Related work

The problem falls in a broader class of geometric optimization problems, where the goal is to find some object inscribed in another object. The optimization measure can differ. During the last 40 years, many such measures have been studied: computing the largest-area convex polygon that is inscribed in a simple polygon [8], the largest-area axis-parallel rectangle that is inscribed in a simple polygon [9], the approximate largest-area rectangle that is inscribed in a convex polygon [10], the largest-area k-gon that is inscribed in a convex polygon [6], the largest-area square and equilateral triangle that are inscribed in a convex polygon [11] and the largest-area parallelogram that is inscribed in a convex polygon [12].

When the input is an (unstructured) set of points in the plane, rather than a convex polygon, we may ask a similar question: what is the largest-area triangle or k-gon that uses only points of the given set as vertices? Clearly, we can attack this problem by first computing the convex hull of the given points. However, this takes $O(n \log n)$ time, and indeed Drysdale and Jaromczyk show that computing the largest-area or largest-perimeter k-gon must take at least $\Omega(n \log n)$ time for any $k \geq 2$, using a reduction from set disjointness [13].

1.2. Contribution

In this paper, we obtain the following results.

- We present a 9-vertex polygon on which the algorithm by Dobkin and Snyder for computing the largest-area triangle [1] fails. By extensions, the algorithm by Boyce *et al.* and Aggarwal *et al.* for computing the largest-area k-gon [6, 7] also fail. In particularly, the example disproves Lemma 2.5 Dobkin and Snyder [1] and Lemma 3.2 of Boyce *et al.* [6] (Section 3).
- We analyse the geometry and present insight into the reason of the failure. We then use this insight to give a quadratic-time algorithm for computing the largest-area triangle in the same spirit as Dobkin and Snyder's algorithm, which we prove is correct (Section 5).
- We then extend our analysis significantly and present a divide-and-conquer algorithm that works in $O(n \log n)$ time (Section 6).

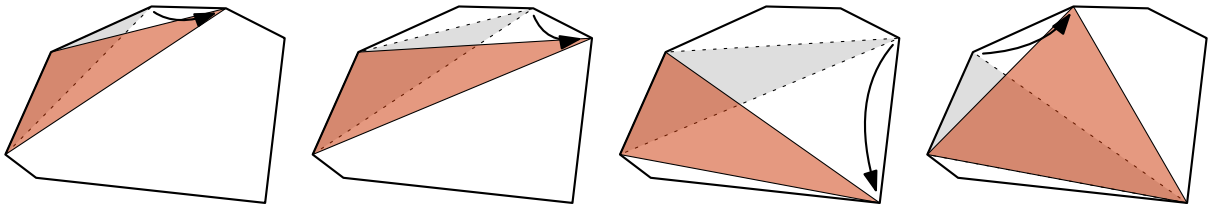


Figure 2: The first four steps of Algorithm 1.

Because the problem is central in computational geometry, a number of follow-up results that depend on largest-area triangles or k-gons, either directly by using the algorithm as a preprocessing step or indirectly by relying on false claimed properties, will have to be reevaluated [1, 14, 15, 7].

2. Preliminaries

In this section, we review several core concepts introduced by Dobkin and Snyder [1] and Boyce *et al.* [6], and describe the algorithm by Dobkin and Snyder in detail.

2.1. Definitions

We first start with some definitions. Let P be a given convex polygon with n vertices. We say a convex polygon Q is *P-aligned* if the vertices of Q are a subset of the vertices of P. Note that there always exists a P-aligned largest-area k-gon inscribed in P (assuming $k \leq n$). Boyce *et al.* [6] define a *rooted* polygon Q with root $r \in P$ to be any P-aligned polygon that includes r. Let P be a convex polygon. Two P-aligning polygons are said to *interleave*, if between every two successive vertices of one, there is a vertex of the other (possibly coinciding with one of them) [6]. Dobkin and Snyder [1] define a *stable* triangle to be a rooted triangle $T = pqr$, where r is the root, such that any other P-aligned triangle $p'qr$ or $pq'r$ has smaller area than T. Henceforth, we will refer to such triangles as *2-stable*. We also define a *3-stable* triangle to be an (rooted or unrooted) P-aligned triangle $T = pqr$ such that any other P-aligned triangle $p'qr$ or $pq'r$ or pqr' has smaller area than T. Note that in degenerate cases, there could be multiple triangles with equal area. If two vertices of P are candidates for the third vertex of a (2 or 3)- stable triangle, the last one in the given cyclic order on P would be accepted. With this assumption, all the possible (2 or 3)-stable triangles on a given root are unique. Note that the largest-area triangle is 3-stable.

2.2. Dobkin and Snyder's triangle algorithm

We will now recall the *triangle algorithm* [1], outlined in Algorithm 1 and illustrated in Figure 2.

Algorithm 1: Triangle algorithm

Input P: a convex polygon, r: a vertex of P
Output T: a triangle
a = r
b = `next`(a)
c = `next`(b)
m = 0
while $a \neq r$ **do**
  if $\triangle abnext(c) > \triangle abc$ **then**
    c = `next`(c)
  end
  else if $\triangle anext(b)c > \triangle abc$ **then**
    b = `next`(b)
  end
  else
    a = `next`(a)
  end
  if $\triangle abc > m$ **then**
    m = $\triangle$abc
  end
end
return m

Let $P = \{p_0, p_1, \ldots, p_{n-1}\}$. Assume an arbitrary vertex of P is the root, assign this vertex and its two subsequent vertices in the cyclic order on the boundary of P to variables a, b and c. We then "move c forward" along P as long as this increases $\triangle abc$. (By $\triangle$ we mean the area of the triangle). If we can no longer advance c, we advance b if this increases $\triangle abc$, then try again to advance c. If we cannot advance either b or c any further, we advance a. We keep track of the largest-area triangle found, and stop when a returns to the starting position. Since a visits n vertices and b and c each visit fewer than $2n$ vertices, the algorithm runs in $O(n)$ time if we have the cyclic ordering of the points on P.

2.3. Correctness

Dobkin and Snyder [1] claim that Algorithm 1 computes the largest-area triangle inscribed in P. Their argument hinges on the following key lemma.

Lemma 1 ([1, Lemma 2.5])**.** *Let $P = \{p_0, p_1, \ldots, p_{n-1}\}$ be a convex polygon. There exists an i ($0 \leq i \leq n-1$) such that the p_i-anchored maximum triangle is the largest-area triangle inscribed in P.*

We claim that Lemma 2.5 [1] is false. In the lemma, the *p_i-anchored maximum triangle* refers to the largest triangle found by the algorithm while $a = p_i$. Note that this is not necessarily the same as the largest-area rooted triangle at p_i. This essential observation lies at the heart of the following construction.

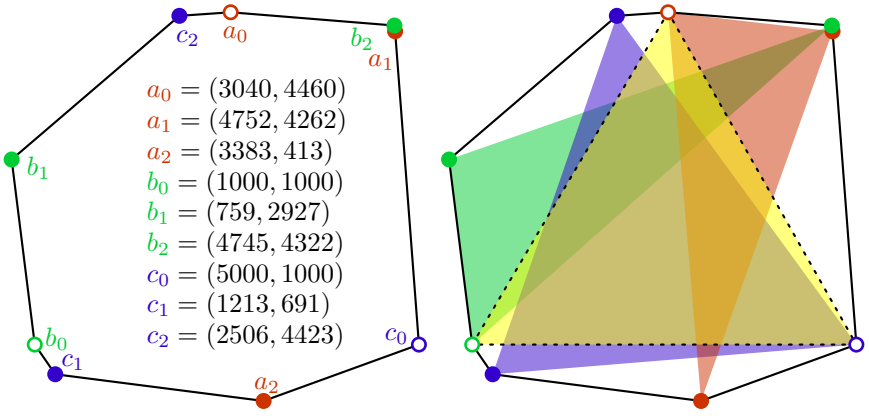


Figure 3: (left) A polygon on 9 vertices. (right) Triangles $a_0a_1a_2$ (red), $b_0b_1b_2$ (green) and $c_0c_1c_2$ (blue) are all 2-stable, but smaller than triangle $a_0b_0c_0$ (yellow).

3. Counter example

In Figure 3 we provide a polygon P on 9 vertices such that the largest-area inscribed triangle and the triangle computed by Algorithm 1 are not the same. We use the following points: $a_1 = (4752, 4262), a_2 = (3383, 413), b_1 = (759, 2927), b_2 = (4745, 4322), c_1 = (1213, 691), c_2 = (2506, 4423), a_0 = (3040, 4460), b_0 = (1000, 1000), c_0 = (5000, 1000)$. The largest-area triangle is $\triangle a_0b_0c_0$; however, Algorithm 1 reports triangle $\triangle c_0c_1c_2$ as the largest-area triangle.

4. Some observations on the largest-area triangle

In this section, we argue that Algorithm 1 does work on convex polygons in which there exists only one 2-stable triangle per vertex. However, if there are multiple 2-stable triangles rooted at the same vertex, the algorithm only works if the first such triangle considered happens to be the largest one. Then we analyze the number of 3-stable triangles.

4.1. The number of 2-stable triangles

We first make some observations and bound the maximum number of 2-stable triangles.

Lemma 2. *Let r be any vertex of P. All 2-stable triangles rooted at r are interleaving.*

Proof. Suppose that the lemma is false. Figure 4 illustrates the proof. Suppose P is a convex polygon and a is a vertex on P with several 2-stable triangles. So there exists at least one 2-stable triangle abc rooted at a that is not interleaving with at least another 2-stable triangle $ab'c'$.

First suppose the case where c occurs before c' but b' occurs before b in counter-clockwise order. Consider lines L and L' through b that are parallel to the supporting lines ac and ac' respectively. As $\triangle ab'c'$ is 2-stable, b' must lie under or on L'; by assumption b' must be

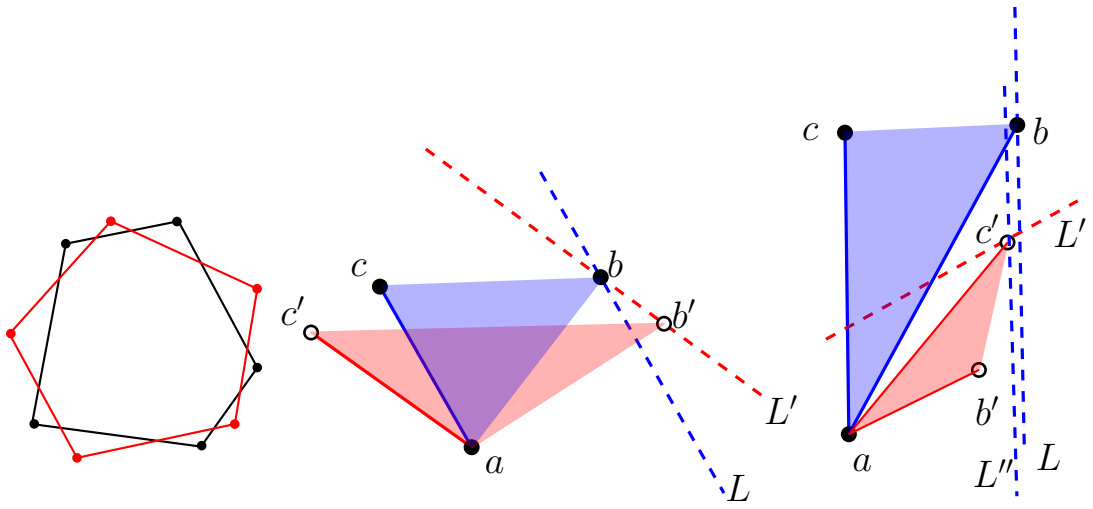


Figure 4: (left) Interleaving polygons. (right) Stable triangles sharing a common root are interleaving.

located before b. But with all of the possible candidates for b', abc would not be longer a 2-stable triangle, contradicting the assumption that abc is a 2-stable triangle.

Second suppose the case where b' and c' occur before b and c in counter-clockwise order. Consider the line L through b that is parallel to the supporting line ac. As $\triangle abc$ is 2-stable, both of b' and c' must lie on the left of L and on the right of supporting line of ab. Consider lines L' and L'' through c' that are parallel to the supporting line ab' and L respectively. By assumption $\triangle abc$ is 2-stable, b must be located to the right of L'' and to the left of L', but then b would not occur after c', contradicting the assumption that b' and c' occur before b and c. □

Observation 1. *The number of 2-stable triangles rooted at any given vertex of a convex polygon is at most $O(n)$.*

It is possible that for a given vertex p_0 on a convex polygon P, each edge p_0p_i is an edge of a 2-stable triangle, as illustrated in Figure 6. Furthermore, Figure 5 shows that any of these triangles could be the largest one; that is, the sequence of areas of the 2-stable triangles rooted at p_0 is not necessarily increasing or decreasing. However, p_0p_i can participate in at most two 2-stable triangles, namely, using the vertices that are farthest from the line through p_0 and p_i. So, the number of 2-stable triangles rooted at any vertex of P is $O(n)$.

Furthermore, we can slightly alter the polygon in Figure 5 so that it becomes a polygon P' with $O(n^2)$ 2-stable triangles, see Figure 5. If we replace p_0 with n new vertices $p_0, p_{0_1}, p_{0_2}, \ldots, p_{0_n}$, all on the boundary of P and close to each other, and such that the other points p_i are far enough from each other, the resulting polygon P' can have $O(n^2)$ 2-stable triangles.

Implicitly, Algorithm 1 is based on the assumption that there is only a linear number of 2-stable triangles that are comparable in size and to the size of the largest-area triangle. We can alter the example in Figure 5(right) to make any of the $O(n^2)$ 2-stable triangles the

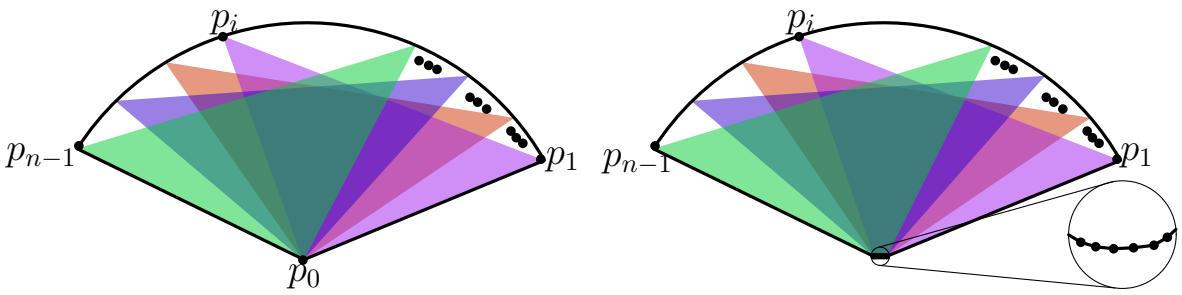


Figure 5: General class of the polygon triangle algorithm [1] fails.

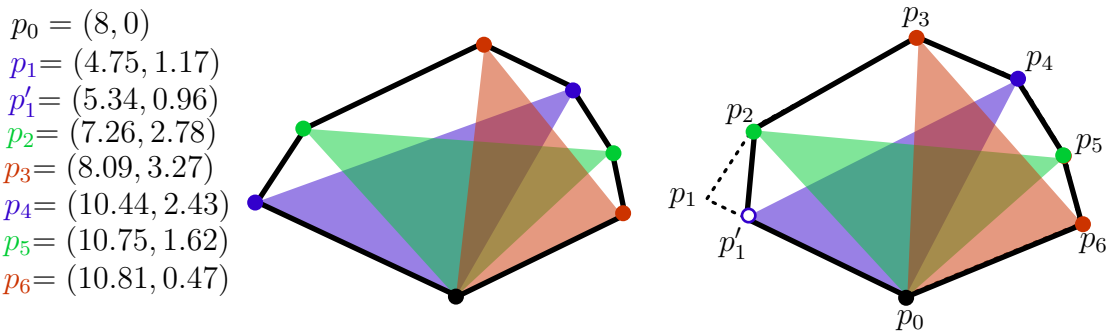


Figure 6: A polygon with three 2-stable triangles on a vertex; the blue triangle is the largest. If we move one vertex, there are still three 2-stable triangles, but now the blue triangle is the smallest.

largest, but independent of such a change, Algorithm 1 will always report $\triangle p_0 p_1 p_i$ as the largest-area triangle.

Clearly, Observation 1 shows that the total number of 2-stable triangles is at most quadratic.

Corollary 2.1. *The total number of 2-stable triangles on a convex polygon is bounded by* $O(n^2)$.

4.2. The number of 3-stable triangles

We first start with some definitions. Let $\triangle abc$ be a 3-stable triangles inscribed in a given convex polygon P. Then let $\triangle a'b'c'$ be a bigger triangle containing $\triangle abc$, such that $a'b'$ is parallel to ab and passes through c, $a'c'$ is parallel to ac and passes through b, and $b'c$' is parallel to bc and passes through a (see Figure 7). Now, $\triangle abc$ decomposes $\triangle a'b'c'$ into four triangles. Each of these triangles other than $\triangle abc$ is called a *containing triangle*. $\triangle abc$ divides the boundary of P into three chains, such that each chain is located inside a containing triangle of $\triangle a'b'c'$.

It should be noted a vertex of a convex polygon can also be part of multiple 3-stable triangles. For instance, the example in Figure 3 has two 3-stable triangles that include c_0.

Finding an upper bound on the number of 3-stable triangles is important. We will show that the number of 3-stable triangles is bounded by $O(n)$, which could give hope that the largest-area triangle problem still admits a linear-time algorithm.

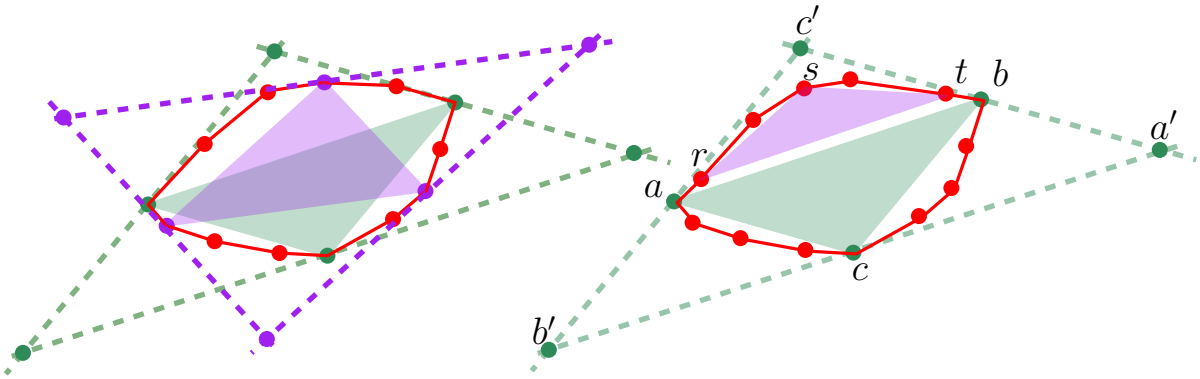


Figure 7: (left) A 3-stable triangle decomposes the boundary of the containing polygon into three chains. (right) 3-stable triangles are intersecting.

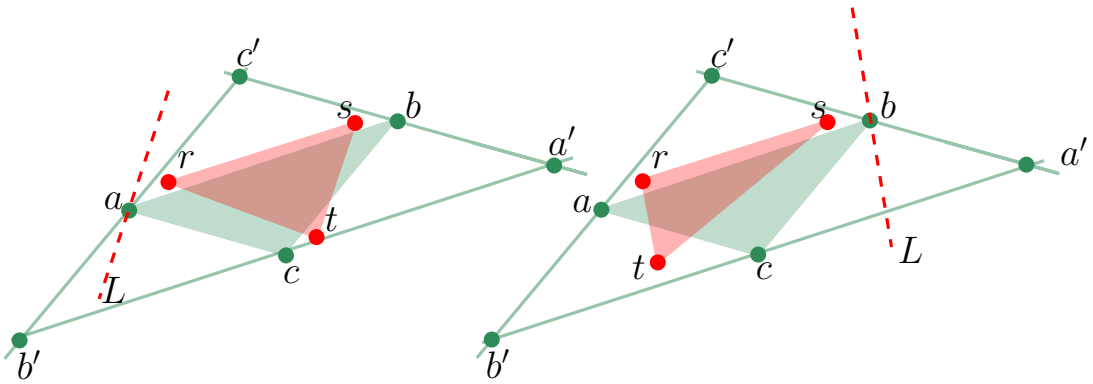


Figure 8: 3-stable triangles are interleaving.

Lemma 3. *Two 3-stable triangles on a convex polygon are always intersecting.*

Proof. We only need to consider 3-stable triangles with disjoint vertices, because 3-stable triangles sharing a root are 2-stable triangles on a common root and are interleaving by Lemma 2 and thus also intersecting. Suppose the lemma is false. Then there exist at least two 3-stable triangles $\triangle abc$ and $\triangle rst$ inscribed in a convex polygon P that are disjoint. Suppose both of $\triangle abc$ and $\triangle rst$ are directed clockwise. But then as all six vertices of both triangles should be on the convex hull, rt must be the neighboring chord of ab. As $\triangle abc$ and $\triangle rst$ are disjoint, $\triangle rst$ should be located inside one of containing triangles of $\triangle abc$, and also $\triangle abc$ should be located inside one of the containing triangles of $\triangle rst$. It is possible if and only if a coincides with r and b coincides with t, contradiction with the disjointness and 3-stable assumption (see Figure 7(right)). □

Lemma 4. *Two 3-stable triangles on a given convex polygon are always interleaving.*

Proof. We only need to consider 3-stable triangles with disjoint vertices. From Lemma 3 previous lemma we know two 3-stable triangles with disjoint vertices are always intersecting. So, suppose we have two 3-stable triangles $\triangle abc$ and $\triangle rst$ inscribed in a convex polygon P

that are not interleaving. It means the vertices of $\triangle rst$ are located only in two containing triangles of $\triangle abc$, otherwise they would be interleaving. Without loss of generality, suppose r and s are both located in $\triangle abc'$ and t is located in $\triangle a'bc$. Let L be the line parallel to st and through r. But then as t is located in $\triangle a'bc$ and s is located in $\triangle abc'$ the slope of L is always greater than slop of supporting line of $b'c'$. But then r should be located outside $\triangle abc'$ to make $\triangle rst$ a 3-stable triangle, contradiction (see Figure 8 (left)). □

Corollary 4.1. *The total number of 3-stable triangles on a given convex polygon is bounded by $O(n)$.*

5. A quadratic-time triangle algorithm

In this section, we present a quadratic-time algorithm to find the largest-area inscribed triangle. Let $P = \{p_0, p_1, \ldots, p_{n-1}\}$ be a given convex polygon. Recall that the largest-area triangle is 3-stable. The idea of the algorithm is to find all 3-stable triangles in P: for each vertex p_i we find all 2-stable triangles rooted at p_i in a single linear pass; because all 3-stable triangles are also 2-stable for some vertex, we find all 3-stable triangles.

Algorithm 2: Quadratic-time triangle algorithm

```
Input P = p_0, ..., p_n: a convex polygon, p_0: a vertex of P
Output Λ: Maximum-area triangle
a = p_0
b = next(a)
c = next(b)
max = 0
while a ≠ p_0 do
    while c ≠ a do
        if △abnext(c) > △abc then
            c = next(c)
        end
        else
        end
        b = next(b)
        if △abc > max then
            max = △abc
        end
    end
    a = next(a)
    b = next(a)
    c = next(b)
end
return max
```

In step i of Algorithm 2, we let $a = p_i$ be the root, and start searching from a and its two subsequent vertices b and c on P. Other than in Algorithm 1, each time we move a,

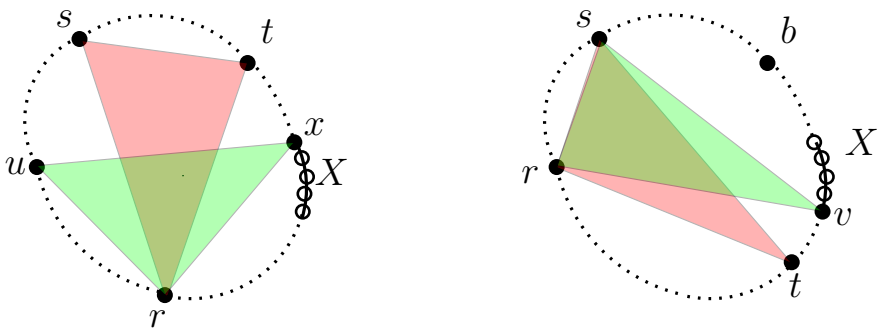


Figure 9: All the rooted 2-stable triangles are interleaving.

we reset b and c, but just like in Algorithm 1, each time we move b, c stays where it is. This means each step of the algorithm now takes linear time, and the total algorithm takes quadratic time.

5.1. Correctness

We will start the correctness proof of Algorithm 2 by the following lemma.

Lemma 5. *Algorithm 2 will consider all 2-stable triangles.*

Proof. Suppose the lemma is false. Then, there exists at least one 2-stable triangle rst rooted at r that the algorithm cannot find when $a = r$. First assume that during the algorithm, at some point, b will reach s. While b is at s, c will traverse some sequence X of vertices of P; let x be the first vertex of X. If t is not in X, there are two cases: t comes before X, or t comes after X.

If t comes before X, then c already passed t before b reached s. This means that b was at some point u when c passed t. But now, $\triangle rux$ and $\triangle rst$ would both be 2-stable, but not interleaved, contradicting Lemma 2 (see Figure 9(left)).

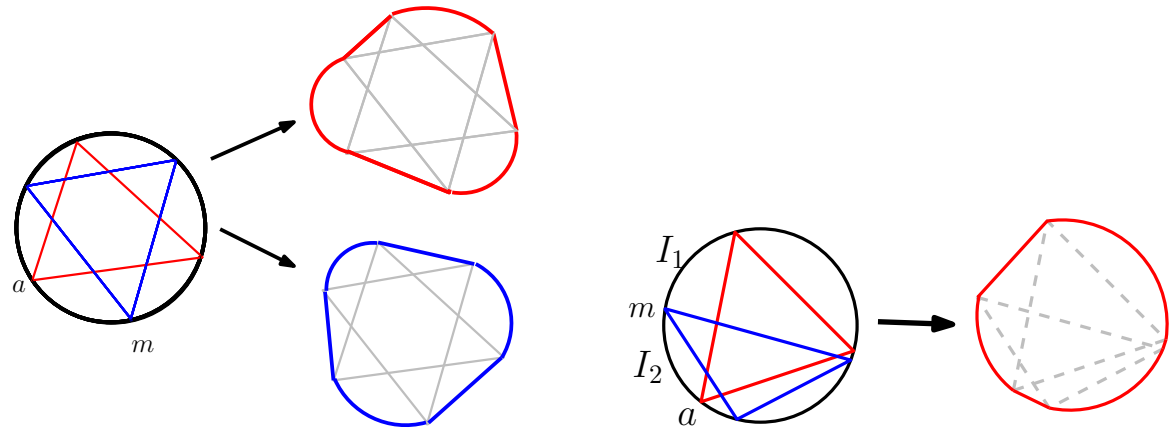
If t comes after X, then b already moved away from s before c reaches t, say, when c was at another vertex v. But then, $\triangle rsv$ was 2-stable. However, $\triangle rst$ is also 2-stable, contradicting the definition of 2-stability.

Now suppose we missed $\triangle rst$ because b did not reach s before c reaches $a - 1$. But then, c passed both s and t, so when this happens $\triangle rbc$ and $\triangle rst$ are not interleaved, a contradiction with Lemma 2 (see Figure 9(right)). □

Theorem 6. *Algorithm 2 will find the largest-area triangle in $O(n^2)$ time.*

Proof. The correctness of the algorithm depends on three facts. First, the largest-area triangle is always a 3-stable triangle; second, in the above procedure we will find all 3-stable triangles; and third, the set of all 3-stable triangles is a subset of the set of all 2-stable triangles. The correctness of the first and third facts are obvious. In Lemma 2 we proved we will consider all the 2-stable triangles. Thus we can conclude the Algorithm 2 works correctly.

□



(a) Interleaving dividing triangles construct two subproblems.

(b) Non-interleaving dividing triangles construct one subproblem.

Figure 10: Dividing triangles and the resulting subproblems.

6. Divide-and-Conquer triangle algorithm

In this section, we will provide a more efficient algorithm for finding the largest-area inscribed triangle on a convex polygon. We will use the following previously established lemmas.

Lemma 7 ([6, Lemma 2.2])**.** *A globally largest-area k-gon and a largest-area rooted k-gon interleave.*

Lemma 8. *The largest-area rooted triangle can be found in linear time.*

Proof. The largest-area rooted triangle on an arbitrary fixed vertex a of P can be found via one step of Algorithm 2. In the correctness proof of that algorithm we mentioned that we can find all the 2-stable triangles on any given root a in linear time. The largest-area rooted triangle can also be found in linear time. □

6.1. Algorithm

In the first step of the algorithm we choose an arbitrary vertex a on P and compute the largest-area triangle rooted at a. We call this triangle ${T^1}_a$. ${T^1}_a$ decomposes the boundary of P into three intervals that share their endpoints. Let m be the median vertex on the largest of these intervals. We then compute the largest-area rooted triangle on m, ${T^1}_m$. We call ${T^1}_a$ and ${T^1}_m$ *dividing triangles*. The vertices of ${T^1}_a$ and ${T^1}_m$ subdivide P into six intervals that share their endpoints; Figures 10a and 10b show two possible configurations.

Recall that, by Lemma 7, the largest-area triangle Λ we are looking for must interleave with both ${T^1}_a$ and ${T^1}_m$. This implies that, once we fix the interval that contains one vertex of Λ, the other two vertices are constrained to lie in two pairwise disjoint intervals. Depending on the configuration, there could be either one or two sets of three compatible intervals; if there are two, they must interleave. As a result, we construct either one or two smaller polygons P' or P' and P'' by directly connecting these intervals.

In the second step, we will repeat the above procedure again by finding another largest triangle T_a^2 rooted on an arbitrary vertex a of P' (or P''), and another largest triangle T_m^2 rooted on the median vertex m of the largest sub-interval of P' (or P'') induced by T_a^2.

Recursively repeating this, we show that for some subproblem $P^\star$ in step i of the algorithm (note that in step i there may be up to 2^i separate subproblems) if ${T^i}_a$ and ${T^i}_m$ are interleaving triangles in $P^\star$, we decompose the problem into two smaller subproblems, and if they are not interleaving, we get a single smaller subproblem. In all cases, the size of each subproblem is between $\frac{1}{6}$ and $\frac{5}{6}$ times the size of the previous subproblem, and in the case where we have two subproblems, the sum of their sizes equals the size of the previous subproblem plus 6.

We will repeat the procedure of constructing dividing triangles in each step on one or two smaller polygons, until our subproblems become triangles themselves; in this case we simply return the area of the triangle. This procedure is outlined in Algorithm 3.

Algorithm 3: Divide-and-Conquer triangle algorithm

```
Procedure LARGEST-TRIANGLE(P)
Input P: a convex polygon
Output The largest-area triangle in P
if |P| = 3 then
    return P
end
else
    a = an arbitrary root on P
    T_a = largest-area triangle rooted at a
    m = median point on the largest interval on P between two vertices of T_a
    T_m = largest-area triangle rooted at m
    P',P''=sub-polygons constructed by interleaving intervals using T_a and T_m
    if T_a and T_m are interleaving then
        return max (LARGEST-TRIANGLE(P'), LARGEST-TRIANGLE(P''))
    end
    else if P' is the sub-polygon can include the largest-area triangle then
        return LARGEST-TRIANGLE(P')
    end
    else
        return LARGEST-TRIANGLE(P'')
    end
end
```

6.2. Time complexity

We start analyzing the time complexity of the algorithm with the following lemma.

Lemma 9. *Let P be a convex polygon with n vertices. The (one or two) subproblems induced by P have size at most $\frac{5}{6}(n+6)$.*

Proof. The dividing triangles T_a and T_m decompose the boundary of P into six intervals. Let I_1 and I_2 be the two intervals incident to m. Only one of I_1 or I_2 can include a vertex of the largest-area triangle, otherwise the largest-area triangle would no longer interleave both T_a and T_m. Consider the $n-6$ vertices not part of T_a or T_m. Because of the choice of m, both I_1 and I_2 contain at least a factor $\frac{1}{6}$ of these vertices, so $\frac{1}{6}(n-6)$ each (see Figure 10b). Since one subproblem does not contain I_1, and the other subproblem does not contain I_2, each subproblem has size at most $n - \frac{1}{6}(n-6) = \frac{5}{6}(n+6)$. □

Note that, if P splits into two subproblems P' and P'', then $|P'| + |P''| \leq n + 6$.

So the recursive equation of the divide-and-conquer algorithm is

$$T(n) = \max\{T(\alpha(n+6)) + T((1-\alpha)(n+6)) + O(n+6), T(\alpha(n+6) + O(n+6)\}$$

where $\frac{1}{6} \leq \alpha \leq \frac{5}{6}$, by Lemma 9.

By using $T(3) = 1$ and considering the maximum part of the above equation, and using the method of Akra and Bazzi [16], the recursion can be written as

$$T(m) = T(\alpha m) + T((1-\alpha)m) + m$$

and as m is in $O(n)$, $T(n)$ is bounded by $O(n \log n)$.

6.3. Correctness

For the correctness proof of Algorithm 3 we will show that in each step of the recursion we always transfer all three vertices of the largest-area triangle Λ to the same subproblem (or to both subproblems, if and only if $\Lambda = T_a$ or $\Lambda = T_m$).

Lemma 10. *In each step of Algorithm 3, there is always at least one subproblem containing all three vertices of the largest-area triangle* Λ.

Proof. Let P be a convex polygon, and consider the dividing triangles T_a and T_m. If $\Lambda = T_a$ or $\Lambda = T_m$, clearly both subproblems of P contain all three vertices of Λ.

Otherwise, consider a subproblem P' and suppose it does not contain all three vertices p, q, and z of Λ; say (w.l.o.g.) P' contains p but not z. We know $\Lambda = \triangle pqz$ must interleave both T_a and T_m. But then z must be in P'. This is only possible if z is a vertex of T_a or T_m. □

Theorem 11. *Algorithm 3 will find the largest-area triangle in* $O(n \log n)$ *time.*

References

References

[1] D. P. Dobkin, L. Snyder, On a general method for maximizing and minimizing among certain geometric problems, in: 20th Annual Symposium on Foundations of Computer Science (sfcs 1979), 1979, pp. 9–17. doi:10.1109/SFCS.1979.28.

[2] M. I. Shamos, Problems in computational geometry, unpublished manuscript, 1975.
[3] M. I. Shamos, Computational geometry. ph.d. thesis. (May 1978) 1978.
[4] G. T. Toussaint, Solving geometric problems with the rotating calipers, in: Proc. IEEE Melecon, Vol. 83, 1983, p. A10.
[5] D. Avis, G. T. Toussaint, B. K. Bhattacharya, On the multimodality of distances in convex polygons, Computers & Mathematics with Applications 8 (2) (1982) 153 – 156. doi:http://dx.doi.org/10.1016/0898-1221(82)90054-2.
URL http://www.sciencedirect.com/science/article/pii/0898122182900542
[6] J. E. Boyce, D. P. Dobkin, R. L. S. Drysdale, L. J. Guibas, Finding extremal polygons, SIAM Journal on Computing 14 (1) (1985) 134–147. arXiv:http://dx.doi.org/10.1137/0214011, doi:10.1137/0214011.
URL http://dx.doi.org/10.1137/0214011
[7] A. Aggarwal, M. Klawe, S. Moran, P. Shor, R. Wilber, Geometric applications of a matrix searching algorithm, in: Proceedings of the second annual symposium on Computational geometry, ACM, 1986, pp. 285–292.
[8] J. S. Chang, C. K. Yap, A polynomial solution for the potato-peeling problem, Discrete & Computational Geometry 1 (2) (1986) 155–182. doi:10.1007/BF02187692.
URL http://dx.doi.org/10.1007/BF02187692
[9] K. Daniels, V. Milenkovic, D. Roth, Finding the largest area axis-parallel rectangle in a polygon, Computational Geometry 7 (1) (1997) 125 – 148. doi:http://dx.doi.org/10.1016/0925-7721(95)00041-0.
URL http://www.sciencedirect.com/science/article/pii/0925772195000410
[10] S. Cabello, O. Cheong, C. Knauer, L. Schlipf, Finding largest rectangles in convex polygons, Computational Geometry 51 (2016) 67–74.
[11] A. DePano, Y. Ke, J. O'Rourke, Finding largest inscribed equilateral triangles and squares, in: Proc. 25th Allerton Conf. Commun. Control Comput, 1987, pp. 869–878.
[12] K. Jin, K. Matulef, Finding the maximum area parallelogram in a convex polygon, in: CCCG, 2011.
[13] R. L. S. Drysdale, J. W. Jaromczyk, A note on lower bounds for the maximum area and maximum perimeter k-gon problems, Information processing letters 32 (6) (1989) 301–303.
[14] V. Keikha, M. Löffler, A. Mohades, Largest and smallest area triangles on a given set of imprecise points, in: EuroCG, 2017.
[15] B. Bhattacharya, H. ElGindy, A new linear convex hull algorithm for simple polygons, IEEE Transactions on Information Theory 30 (1) (1984) 85–88.
[16] M. Akra, L. Bazzi, On the solution of linear recurrence equations, Computational Optimization and Applications 10 (2) (1998) 195–210.